

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

Lecture - 44
El Gamal Public Key Encryption Scheme

(Refer Slide Time: 00:33)

Roadmap

□ El Gamal Public-key encryption scheme

❖ CPA-security

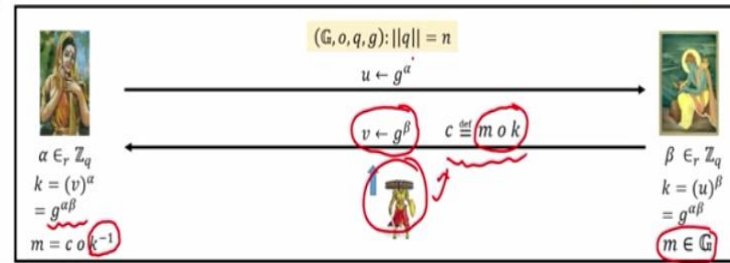
❖ Implementation issues

Hello everyone. Welcome to this lecture. Just to recap, in the last lecture, we have seen the syntax of public-key encryption scheme. So the roadmap for this lecture is as follows. In this lecture, we will see a candidate public-key encryption scheme, namely El Gamal public-key crypto system and we will prove formally its CPA security and we will end the lecture with some of the implementation issues, which we face while implementing El Gamal encryption scheme in practice.

(Refer Slide Time: 00:56)

El Gamal Encryption Scheme : Intuition

Recall the DH key-exchange protocol --- for simplicity, consider a **multiplicative cyclic group**



Consider the following **encryption process**:

- ❖ Sender and receiver runs an instance of DH key-exchange protocol to obtain a **shared key** $k \in \mathbb{G}$
 - If **DDH assumption holds** then k is indistinguishable from any random element element of \mathbb{G}
- ❖ Sender can use k to "**mask**" its plaintext m --- receiver can "**unmask**" k from the ciphertext

Claim : Distribution of $(m \circ k)$ is independent of the underlying plaintext m

So let us try to understand the intuition of El Gamal encryption scheme. So for that, recall the Diffie–Hellman key-exchange protocol. And for simplicity, assume we are considering a multiplicative cyclic group. So the public parameters are the description of a cyclic group, a generator and the size of the group, which is q . And in the Diffie–Hellman key-exchange protocol, say Sita and Ram, they want to agree upon a key, each of them pick up their own contribution for the overall key. So Sita picks her contribution α , and she sends g^α to Ram. And independently, Ram picks his contribution β and sends g^β . And overall key k that is agreed between the sender and receiver is $g^{\alpha\beta}$. And we had formally proved the security of the Diffie–Hellman key exchange protocol.

So now, consider the following encryption process. So sender and receiver first runs an instance of the Diffie–Hellman key-exchange protocol to obtain a shared key, denoted by k , which is a group element. And we know that if the DDH assumption holds in the underlying group, that means if the DDH problem is difficult to solve in the underlying group, then the agreed key k is indistinguishable from any random element of the group.

Now imagine, Ram has a message, say plain text m , which is a group element, which it wants to encrypt and send it to Sita. So what Ram can do is, from the view point of Ram, Ram knows that by running the Diffie–Hellman key exchange protocol, Sita is also going to have the same key k . And Ram also knows that if there is an eavesdropper, who has eavesdropped the communication

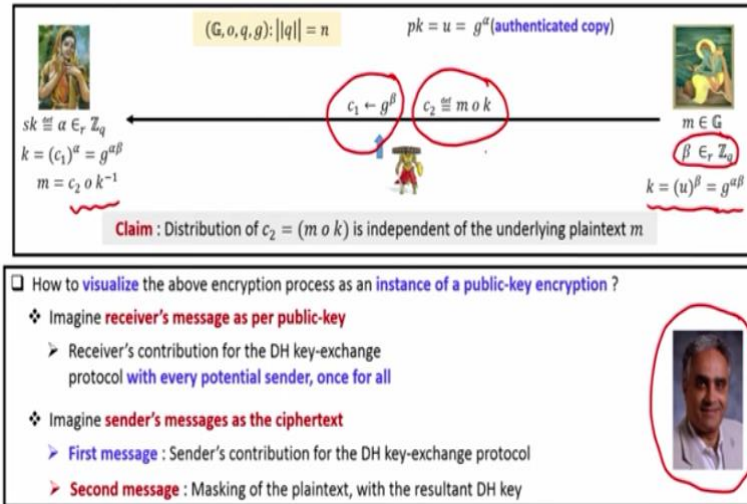
between Sita and Ram, then from the view point of that adversary, the key k , which is available with Ram, is kind of indistinguishable from any random element from the group. So what Ram can do is, to encrypt the plain text m , it can use the key to mask its plain task and since we are performing operations in the group, to mask the plain text, what Ram can do is, it can perform the group operation on the message and the key k and the result is denoted by c , which is also sent with the message, which Ram would have sent as part of the Diffie–Hellman key exchange protocol.

Now once Sita receives the messages from Ram, she is now receiving two elements from the group. The first element is Ram's contribution as part of the Diffie–Hellman key exchange protocol, which Sita uses to generate the key k as per the steps of the Diffie–Hellman key exchange protocol. As once it has received the key k , to decrypt the cipher text, what Sita has to do is, she has to just cancel out the effect of key k or she has to unmask the key k . And to unmask the key k , what she can do is, she can just perform the group operation on the cipher text c and the multiplicative inverse of the element k . So since the element k is known to Sita and she knows the group description, she can compute the multiplicative inverse in polynomial time, which I denote by k^{-1} . And if she performs the group operation on the cipher text c and k^{-1} , the effect of k cancels out. And Sita ends up getting the plain text m .

Now, I claim here that the cipher text c , which is the result of the group operation on the plain text m and the key k , is going to be independent of the underlying plain text m . I will prove this very soon, but for the moment assume that this claim is true. If indeed this claim is true, then this whole protocol, this whole process of encryption and decryption indeed looks like a candidate encryption scheme. Because if the distribution of the cipher text c is independent of the key m , then even after seeing the cipher text c , adversary will be unable to figure out what exactly is encrypted in c , whether it is an encryption of m_0, m_1, m_2 , it cannot figure out. So that is the overall intuition of the El Gamal encryption scheme.

(Refer Slide Time: 05:09)

El Gamal Encryption Scheme : Intuition



So I have written the blueprint of the encryption scheme that I had discussed in the last slide. Now the question is that how we can visualize the entire process that we have discussed just now as an instantiation of public-key encryption scheme? Because remember as per the syntax of public-key encryption scheme, we need to have a key-generation algorithm, which would output a public-key, secret-key pair, we should have an encryption algorithm and we should have a decryption algorithm. So pictorially, we know that now we have a blueprint of an encryption process, but now we have to put everything into the syntax of public-key encryption process. And this process of visualization of above encryption process as an instance of public-key encryption scheme was identified by Taher El Gamal. And that is why this encryption process that we are going to discuss now is called as El Gamal encryption scheme.

So you might be wondering that how exactly it is different from this Diffie–Hellman key-exchange protocol? Well, we are not doing anything apart from Diffie–Hellman key exchange protocol. So this part of the communication, which I have highlighted is exactly Diffie–Hellman key exchange protocol. But on top of that, we are doing some additional communication from the sender's side, which allows the receiver to decrypt the cipher text and recover back the plain text. So what we are going to do is, the entire encryption process that we have discussed till now visually, we can imagine it as an instance of public key encryption scheme as follows. So we can imagine that the receiver's message here, namely Sita's message as part of the Diffie–Hellman key-exchange protocol is her public key.

And we can visualize that as if, that is her contribution for the Diffie–Hellman key-exchange protocol with every potential sender, once for all. That means, the key-generation algorithm that Sita can run here is as follows. As part of secret key, she can randomly pick an index α in the range 0 to $q - 1$ and she can make her public key to be g^α . And it will be ensured that it is an authenticated copy. That means, indeed this is g^α generated by so called Sita. How exactly it is ensured, we will see or solve that problem later on. But for the moment assume that Sita has generated a secret key like that and she has computed public key to be g^α and made it available in the public domain. Then, we can imagine as if this is her contribution or her part of the message for the Diffie–Hellman key-exchange protocol with every possible Ram, who would like to do a secure communication with Sita.

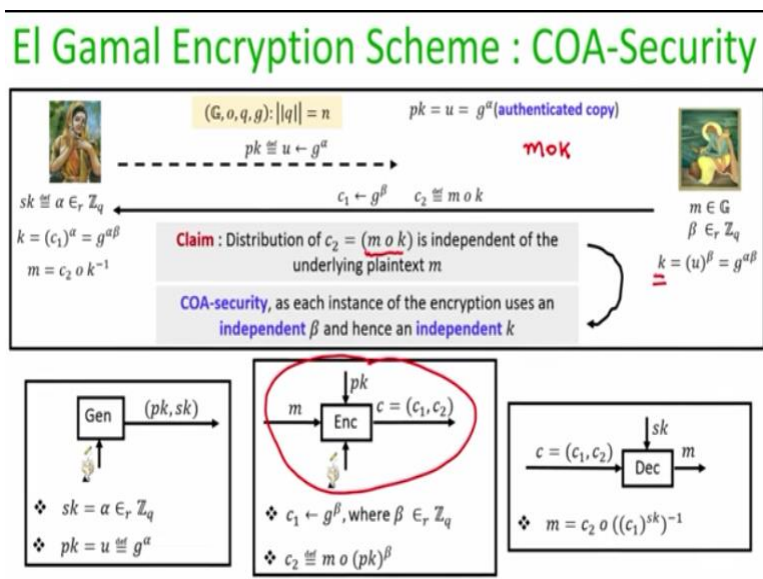
Now, assume we have a so called Ram or a sender who wants to encrypt a plain text, say m , using the public key. So what Ram is going to do is, Ram is going to pick a random β in the range 0 to $q - 1$ and now he is now going to compute two group elements. The first group element is c_1 , which is nothing but g^β . And a second group element c_2 is basically the group operation being performed on the plain text and the key k , where the key k is $g^{\alpha\beta}$, which is obtained by raising the public key u to the index β . So the two messages or the two elements which Ram is sending, can be visualized as follows. The first message, you can interpret as if it is Ram's contributions or sender's contribution for the Diffie–Hellman key-exchange protocol, because if indeed Ram would have participated in an instance of the Diffie–Hellman key-exchange protocol, then c_1 is the message, which Ram would have sent to Sita in response to the message g^α , which Sita has already sent and went offline. The second message c_2 or the second group element c_2 , you can imagine as if it is masking of the plain text with the resultant Diffie–Hellman key, which Sita and Ram would have agreed upon using g^α and g^β as the protocol transcript.

So now if we imagine this encryption process by parsing the messages from Sita and Ram like this, then it automatically fits into the framework of our public-key encryption process. To do the decryption, what Sita has to do is, from the first group element, which Ram has sent, using that and the public key that Sita has sent already to the so called Ram, Sita can perform her steps of the Diffie–Hellman key-exchange protocol and agree upon or retain the same key k , which Ram has

used for masking the message. And once it recovers the key k , to decrypt the cipher text, it takes the second component of the cipher text, namely c_2 and it performs the group operation on c_2 and the multiplicative inverse of k^{-1} to recover back the plain text. My claim here is that in this entire process, the distribution of the second component of the overall cipher text, namely c_2 is independent of the underlying plain text.

So we will soon prove this fact, but now if we imagine this whole thing, like the way I have said, now you can see that we have now an instance of a public-key encryption scheme.

(Refer Slide Time: 10:31)



So now let us put the exact formal details of the El Gamal encryption process. So the plain-text space and the public-key space are both going to be the group. And the secret-key space is going to be \mathbb{Z}_q , namely it is going to be the set $\{0, \dots, q - 1\}$. And overall cipher text will consist of two group elements. So it is going to be a pair of elements from the underlying group. The key-generation algorithm outputs a public key and a secret key as follows.

The secret key is a random α in the range 0 to $q - 1$ and a public key g^α . So that you can imagine as if Sita is doing her part of the Diffie–Hellman key exchange protocol with every potential receiver once for all. The encryption algorithm, which Ram or any sender is going to follow for encrypting a plain text is as follows. The sender is going to pick a random β in the range 0 to $q - 1$. And compute g^β , that is going to be the first component of the cipher text. And the actual

encryption of the message is the group operation performed on the plain text and the public key raised to the power β . So pictorially, you can imagine that first component of the cipher text is nothing but sender's contribution for the Diffie–Hellman key, which sender and receiver are going to agree upon. And the second component of the cipher text is the masking of the plain text with the Diffie–Hellman key.

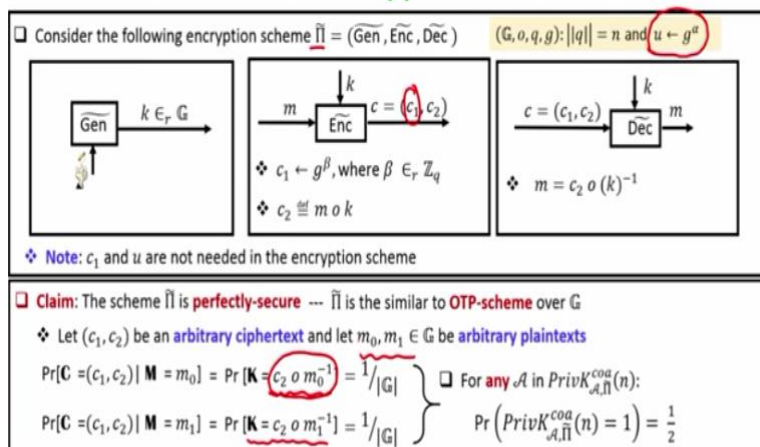
Now the decryption operation is, you receive a cipher text consisting of two group elements. So you first compute a Diffie–Hellman common key, which is going to be established between the sender and the receiver, by raising the group element c_1 to the secret key, so that you also obtain $g^{\alpha\beta}$, find a multiplicative inverse of it. And then perform the group operation with the second component of the cipher text, so that effect of $g^{\alpha\beta}$ vanishes and you end up with the plain text.

So that is the formal syntax of the El Gamal encryption scheme. Now we want to prove formally that this El Gamal encryption scheme is indeed COA secure. As we have discussed in the last lecture, in the public key world, COA security, single message CPA security and multi message CPA security are all equivalent. So it suffices to just prove the COA security of this encryption process. So as I am claiming over the last few slides, the distribution of the cipher text component c_2 , namely the group operation on m with the Diffie–Hellman key k , is going to be independent of the underlying plain text. That means, from the view point of a computationally bounded adversary, if it sees c_2 , then from its view point, c_2 could be the result of applying the group operation on any m and any k . And if that is the case, that means, if this claim is indeed true, then it automatically implies COA security. Intuitively this is because for each instance of the encryption algorithm in this El Gamal encryption scheme, the sender is going to pick a β randomly. It is not the case that it will pick the same β every time. And if β is picked independently for each instance of the encryption, then it automatically means that the Diffie–Hellman key k , which is used for masking the message is also going to be independent for each instance. Because the overall Diffie–Hellman key is $g^{\alpha\beta}$. So even if the α component in the resultant Diffie–Hellman key, which sender and receiver are using to do the encryption and decryption is same, it is the β which is triggering the randomness here and since β is independently picked here, for each instance of the encryption, the overall Diffie–Hellman key k , which is used in each instance is independent.

And now assuming that this claim is true, that means the distribution of c_2 is independent of the underlying plain text, we get the COA security.

(Refer Slide Time: 14:34)

Warm-up : Perfectly-Secure Private-key El Gamal Encryption Scheme



So now let us formalize this intuition by a rigorous proof. And before going into the proof, let us do a warm up here and consider a variation of El Gamal encryption scheme. Mainly we are going to consider a perfectly-secure variant of the El Gamal encryption scheme. I stress here that it is not the way we are going to implement the El Gamal encryption scheme and it is not the way we actually use the El Gamal encryption scheme. This variation of the El Gamal encryption scheme in the private key setting is just to make the proof simpler. So the modified El Gamal encryption scheme in the private key setting, I am denoting as $\tilde{\Pi}$. It has its own key generation algorithm, encryption algorithm and decryption algorithm. The public parameters are the cyclic group, group description and a uniformly random group element g^α , where α is not known.

So we can imagine as if it is some kind of set up, which has been done by a trusted third party and α is not known to anyone. Now since this is a symmetric key encryption process, the key generation algorithm is going to output a uniformly random key and the key is an element of the group. To encrypt a message in this variant of El Gamal encryption process, we compute two group elements, namely c_1 and c_2 . Where c_1 is some g^β , where β is randomly chosen from the set \mathbb{Z}_q .

And cipher text component c_2 is basically the masking of the message with the key k . Since it is a symmetric key encryption scheme, we are going to use the same key k for decryption as well. And to recover the plain text, we basically take the second component of the cipher text and perform the group operation with respect to the multiplicative inverse of the key.

Notice that in this variation of the El Gamal encryption process, the first component of the cipher text, namely c_1 and the publicly known u , they are not at all used for the encryption process and for the decryption process. But I am just retaining them, to ensure that the overall syntax of the cipher text that we are getting here looks like the same as we are going to obtain in the real instantiation of the El Gamal encryption scheme.

Now I claim here that private key variant of the El Gamal encryption process is perfectly secure if my underlying plain text is the group G . And this is because this private key variant of the El Gamal encryption scheme is exactly similar to the one-time pad scheme over the group G . The only difference is that in the one-time pad scheme, we perform the XOR of the key with the plain text. But since we are in the group setting, we are going just replacing that XOR operation by the group operation. More formally, assume that we have an arbitrary cipher text, say (c_1, c_2) and say we consider a pair of arbitrary plain text, namely m_0 and m_1 , which are group elements, because here my plain text space is the underlying group. I am going to show that indeed this encryption process satisfies the definition of perfect secrecy.

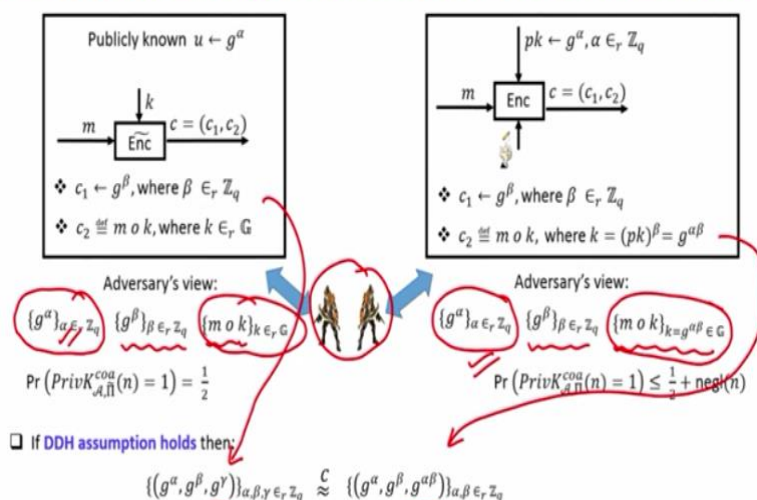
Namely consider the probability that this arbitrary cipher text (c_1, c_2) is an encryption of the plain text m_0 . And in the same way, consider the probability that this arbitrary cipher text (c_1, c_2) is an encryption of m_1 . It turns out that this arbitrary cipher text (c_1, c_2) is an encryption of m_0 , only if key-generation algorithm would have produced a key, which is the result of group operation performed on c_2 and the multiplicative inverse of m_0 . But since the key-generation algorithm outputs uniformly random elements from the group as the key, it turns out that the key generation algorithm indeed outputs a key, which is same as c_2 group operation m_0^{-1} is 1 over the group size. Now by running exactly the same argument, we can claim that the probability that the plain text m_1 is encrypted in the cipher text (c_1, c_2) is exactly the same, that my key generation algorithm outputs a key, which is same as the group operation performed on c_2 and m_1^{-1} .

And the probability that my key is this, is 1 over the size of the group. That means, for any adversary, even if it is computationally unbounded, if it participates in perfectly-secure indistinguishability experiment in the symmetric key setting or the COA experiment, then the probability that it can distinguish apart whether it is seeing an encryption of the group element m_0 or whether it is seeing an encryption of the group element m_1 , is exactly half.

That means, you cannot distinguish apart; with equal probability it is an encryption of m_0 as well as encryption of m_1 . And that is why this modified or symmetric-key variant of the El Gamal encryption process is perfectly secure.

(Refer Slide Time: 19:38)

El Gamal Encryption Scheme : COA-Security



Now let us turn to the COA security of the actual El Gamal encryption scheme that we have designed in the public key setting. So before going further, let us again remember what we have proved just now. So we have considered a variant of El Gamal encryption scheme in the symmetric key setting and here is the encryption algorithm. The encryption algorithm produces two group elements (c_1, c_2) , where c_1 is some random g^β and c_2 is the masking of the message. And apart from that, adversary also have a public information, namely g^α , where α is not known to the adversary. So if I consider the view of the adversary, the adversary's view basically consists of three probability distributions, namely he has an element g^α , where α is randomly chosen from \mathbb{Z}_q .

It knows the value of g^β , where β is randomly chosen from \mathbb{Z}_q . And it knows the masking of the message with the plain text, where the key is chosen randomly from the underlying group. And we have proved that this encryption process is perfectly secure.

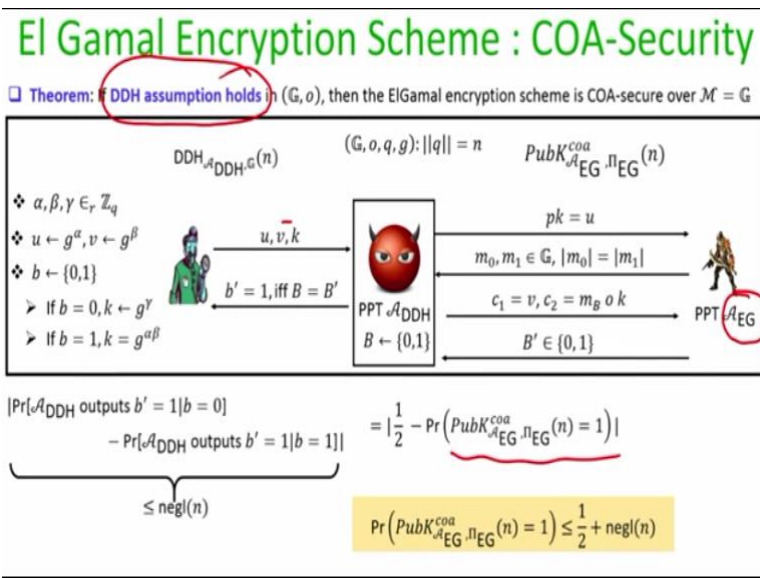
On the other hand, the actual El Gamal public key encryption scheme, that we have designed, there also the cipher text consists of two group elements. And second component of the cipher text is the masking of the message with the Diffie–Hellman key, namely $g^{\alpha\beta}$. So if I consider the adversary's view in this real instantiation or the actual instantiation of the El Gamal encryption scheme in the public key setting, then its view is as follows. It knows the value of g^α , where α is unknown and uniformly random from the set \mathbb{Z}_q . It knows the value of g^β , where β is uniformly random from the set \mathbb{Z}_q . And it knows the masking of the message with the Diffie–Hellman key, where the Diffie–Hellman key is nothing but $g^{\alpha\beta}$, and it belongs to the underlying group.

Now if you see here closely, what exactly is differing here, if I consider the views of the two adversary here? The distribution of g^α in both the worlds or for the adversaries are perfectly the same. They are exactly indistinguishable. Here also alpha is random, here also alpha is random, not known to the adversary and adversary knows the value of g^α . In the same way, the distribution of g^β in both the worlds are exactly identical. What is differing here, is the nature of c_2 that adversary sees in the symmetric key variant of El Gamal scheme and the distribution of c_2 , which adversary sees in the actual El Gamal encryption scheme.

In the symmetric key world, the masking is with a uniformly random group element k , whereas in the public key El Gamal, the masking of the plain text is with the pseudo-random key k , which is a Diffie–Hellman key $g^{\alpha\beta}$. And if I assume that the DDH assumption holds in my underlying group, then we know that as per the Diffie–Hellman assumption, Diffie–Hellman triplet and a non-Diffie–Hellman triplet, they are computationally indistinguishable from the view point of any computationally bounded adversary. That means, if my k is uniformly random, that means if I am in this case, then k in that case is some g^γ , where γ is totally random, not related to α and β . Whereas if I consider cipher text c_2 as per the public key Diffie–Hellman public-key El Gamal encryption scheme, then my key k is nothing but $g^{\alpha\beta}$.

So if my adversary cannot distinguish between a DH triplet and a non-DH triplet, then I can say that the distribution of the cipher text component c_2 , which adversary sees in both the worlds are also computationally indistinguishable and that will automatically prove that our El Gamal encryption process is COA secure.

(Refer Slide Time: 23:45)



So that is the formal statement which we are going to prove now. We are going to prove that if the DDH assumption holds in my underlying group, then the El Gamal encryption process is indeed COA secured. And we formally establish this fact by giving a reduction. So assume, you have a poly time adversary who can attack your El Gamal public key encryption scheme. Using that attack, we are going to design a DDH solver, a poly time DDH solver who can distinguish apart a Diffie–Hellman triplet from a non-Diffie–Hellman triplet.

So it participates in an instance of the DDH experiment. The DDH experiment prepares a challenge for the DDH solver by giving him (u, v, k) , where u and v are random group elements. And third component of the triplet is either $g^{\alpha\beta}$, or it is a uniformly random element g^γ , depending upon whether the challenger has $b = 0$ or $b = 1$. And the task of the DDH solver is to find whether it is a DH triplet or a non-DH triplet.

To solve that, the DDH solver invokes our attacker, who can attack the El Gamal encryption scheme and participates in an instance of the COA game and it sets up the public key to be u . Now as per the rules of the COA game, the COA attacker will submit a pair of challenge plain texts from the underlying group and this DDH solver is going to randomly choose one of those two messages and it prepares the challenge cipher text as follows.

The second component of the triplet, which is given as the challenge to this DDH solver, is set to be the first component of the cipher text and the actual encryption of the message is set as m_b , masked with the third component of the triplet, which is thrown as a challenge to the DDH solver. So now before proceeding further, let us try to understand what is happening in this overall reduction.

If you see here that if this triplet is a non-DDH triplet, that means k in this case is some g^γ , where γ is not related to α and β , then the distribution of the cipher text (c_1, c_2) , which is given to this attacker against the El Gamal scheme, has exactly the same distribution if this attacker would have participated in the COA game against the symmetric-key variant of the El Gamal encryption scheme. Because that is how this challenge cipher text would look like for the attacker in that experiment. Whereas, if the triplet that is given to this DDH solver is a DH triplet, then the distribution of (c_1, c_2) that this adversary is seeing is exactly the same distribution as if this adversary would have seen by participating in an instance of COA game against the El Gamal encryption process. So we will come back to that fact again.

So now this adversary has to identify whether it has seen an encryption of m_0 or m_1 . So it submits its output b' . And response from the DDH solver is that, it says that it is seeing a DH triplet, if and only if the adversary \mathcal{A}_{EG} has correctly identified whether it is m_0 or whether it is m_1 , which is encrypted in the challenge cipher text (c_1, c_2) . So now let us analyse the advantage, namely with how much probability this DDH solver is going to solve a random instance of the DDH problem.

So I claim that if $b = 0$, that means this triplet is a non-DDH triplet. Then the probability that my DDH solver outputs incorrectly, namely it outputs $b' = 1$, is exactly the same with which this COA attacker would have won the COA game against the symmetric-key variant of the El Gamal

encryption scheme. And we have already proved that it is $1/2$. This is because if we are in the setting where $b = 0$, then as I have already proved that the cipher text, which our adversary \mathcal{A}_{EG} is seeing, has exactly the same distribution as it would have seen by participating in an instance of COA game against the modified El Gamal encryption scheme.

On the other hand, I claim that if my case is $b = 1$, then the probability that my DDH solver outputs $b' = 1$ is exactly the same that my adversary \mathcal{A}_{EG} wins the COA game against the El Gamal encryption scheme. And this follows from the fact that if we are in the case where $b = 1$, then that means the triplet that is given is a DDH triplet or Diffie–Hellman triplet, which means that the distribution of the cipher text, whichever adversary is seen is exactly the same as distribution of the cipher text that this adversary would have seen by participating in an instance of COA game against the El Gamal encryption scheme.

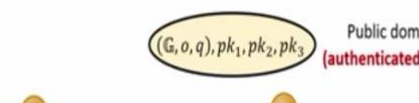
So in summary, what we are concluding now is that, if you see the distinguishing advantage of our DDH solver, then it is exactly $1/2$ minus the probability with which our adversary could have won the COA game against the El Gamal encryption scheme. But since I am assuming that DDH assumption holds in the underlying group, then I know that the distinguishing advantage of any DDH solver is upper bounded by some negligible probability.

That means, if I reshuffle the terms, I end up showing that the advantage of my adversary \mathcal{A}_{EG} in winning the COA game against the actual El Gamal encryption scheme is half plus negligible and hence my El Gamal encryption scheme is indeed COA secure.

(Refer Slide Time: 29:38)

El Gamal Cipher : Implementation Issues

- ❑ **Sharing public parameters:** each instantiation of ElGamal cipher requires the description of (G, o, q)
 - ❖ Multiple receivers can use the same publicly-known (G, o, q) , with each operating with its own sk, pk



- ❖ $sk_1 = \alpha_1 \in_r \mathbb{Z}_q$
- ❖ $pk_1 = u_1 \stackrel{\text{def}}{=} g^{\alpha_1} \in R_1$
- ❖ Sharing **RSA public parameters** among multiple receivers can be insecure

- ❑ **Choice of (G, o, q) :** preference is for **prime-order groups** (DDH problem believed to be **hardest** in such groups)
 - ❖ **Candidate groups:** $(E(\mathbb{Z}_p), +), (G, \cdot_p)$ where G is a prime-order subgroup of \mathbb{Z}_p^*

- ❑ **Message space:** the message space for ElGamal cipher is G , but real-world message space is $\{0, 1\}^*$
 - ❖ **Option I:** Use a **reversible encoding** of bit-strings to group elements (**applicable only for some groups**)
 - ❖ **Option II:** Use ElGamal encryption as a part of **hybrid encryption scheme**

So now let us discuss some of the implementation issues, which we face when we implement El Gamal encryption process. So first thing here is that sharing public parameters in the context of El Gamal encryption scheme is safe. What I mean by that is, if you see, each instantiation of the El Gamal encryption process requires the description of a cyclic group, its generator, group operation and so on. So by sharing public parameters, I mean that multiple receivers can use the same description of the group, the same generator, and so on, with the only difference being that each of them working with their own public key and secret key. That means, if we have scenario where say we have three receivers, R_1, R_2, R_3 , then instead of using different cyclic groups, all the three receivers can operate on the same group, of course by using different public key and secret key.

This is a very remarkable feature with respect to El Gamal encryption process, because later on when we will discuss RSA public-key encryption scheme, in the context of RSA public-key encryption scheme, sharing public parameters, that means say the modulus, exact group on which we are performing the group operation and so on, can lead to insecurity.

Now the second concern here is that the El Gamal encryption scheme that we have discussed is with respect to an abstract cyclic group. But when we are implementing it, we have to select a group, over which we are actually going to perform the operations. So the candidate groups, which we use in practice to instantiate the El Gamal encryption process are as follows. We can either use

the group based on the points on elliptic curves modulo a prime or we can use the prime order multiplicative subgroup of the group \mathbb{Z}_p^* .

So these are two of the popular groups, which we use for instantiating the El Gamal encryption process, because we believe that the DDH problem is indeed hard in both these candidate groups.

The third issue here is the message space. So if you see the description of the El Gamal encryption process, the message space is nothing but the group. But in real world application, we would like to encrypt messages, which are bit strings. So we have now some kind of incompatibility. My actual plain text is a bit string, whereas my encryption process supports elements of groups to be encrypted. So I can remove this incompatibility in either of the two ways. The option one is, we can use some kind of reversible encoding to map bit strings to group elements and vice versa. So this is one way to remove the incompatibility, but this is not preferred.

The second option to get rid of this incompatibility is to use El Gamal encryption process as a part of hybrid encryption scheme. And what I mean by hybrid encryption scheme is that we use the El Gamal encryption scheme just to encrypt a random group element from the sender to the receiver. And then we apply a key derivation function on that encrypted group element, because the same group element will be now decrypted by the receiver and that common group element can serve as a common key for the sender and a receiver.

Now to derive a bit string as a key from that agreed upon group element, both sender and receiver can apply a key-derivation function. And once a key-derivation function is applied, both of them receive a common bit string as a key, which can be now used as a key for a symmetric-key encryption process. So that is what I mean by a hybrid encryption process, because it is a combination of both public key and symmetric-key primitive.

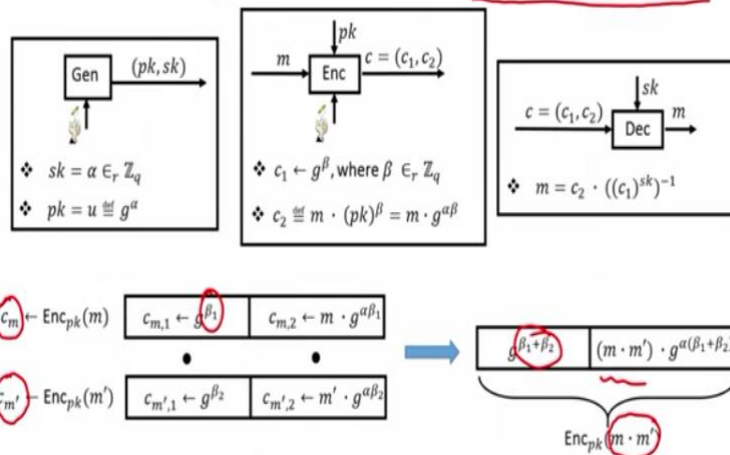
And it will turn out that for most of the public key crypto systems that we are going to discuss, we will face this compatibility issue again and again. And a popular option to deal with this incompatibility issue is to go for this option two, namely use the public key encryption process as

a part of hybrid encryption process. So we will touch upon these details in our subsequent discussions.

(Refer Slide Time: 33:43)

El Gamal Cipher : Multiplicative Homomorphic

□ $\mathcal{PK} = \mathcal{M} = \mathbb{G}$ and $\mathcal{SK} = \mathbb{Z}_q$ and $\mathcal{C} = (\mathbb{G} \times \mathbb{G})$, where (\mathbb{G}, \cdot) is a multiplicative group (for simplicity)



So now let me end this lecture with a very interesting feature of the El Gamal encryption process, by showing that it is multiplicative homomorphic. So I am retaining the description of the El Gamal encryption process. And for simplicity, again I am assuming that my underlying group operation is the multiplication operation, say multiplication modulo p operation. So imagine, I am given with an encryption of some unknown plain text m .

So I do not know the plain text m , but I know the public key and I have an El Gamal encryption of that unknown plain text m , which consists of two group elements, which I am denoting by $c_{m,1}$ and $c_{m,2}$. And as per the syntax of the El Gamal encryption process, $c_{m,1}$ will have this property, so β_1 is the underlying randomness used by the sender. And in the same way, imagine that I have an El Gamal encryption or cipher text of an unknown message m' , again consisting of two group elements. Now suppose I multiply the first component of both the cipher texts. And independently I multiply the second component of both the cipher texts. And this will produce two group elements, which will mathematically have the following property. The first group element will be nothing but g to the power randomness used in the first cipher text plus the randomness used in the second cipher text. And the second component will be the product of the two plain texts, multiplied by g to the power public-key times the summation of the two randomness. And if you

look closely, this is nothing but you can imagine as if this is an El Gamal cipher text for the plain text $m \cdot m'$, under the randomness $\beta_1 + \beta_2$. And that is why I say that my El Gamal encryption process is multiplicative homomorphic.

The reason it is multiplicative homomorphic is that if I multiply two El Gamal cipher texts, then even without knowing the underlying plain texts (I stress I do not know the underlying plain texts and underlying randomness β_1 and β_2 , which are used individually), I end up getting an El Gamal cipher text of a related plain text, namely $m \cdot m'$, under some unknown randomness, namely $\beta_1 + \beta_2$. So this is kind of a very interesting property of El Gamal encryption process. And later on when we will discuss the CCA security of public key encryption process, namely El Gamal encryption process, we will come back and take this property again.

So that brings me to the end of this lecture. In this lecture, we have seen a candidate CPA secure public key encryption process, namely El Gamal encryption process. Thank you.