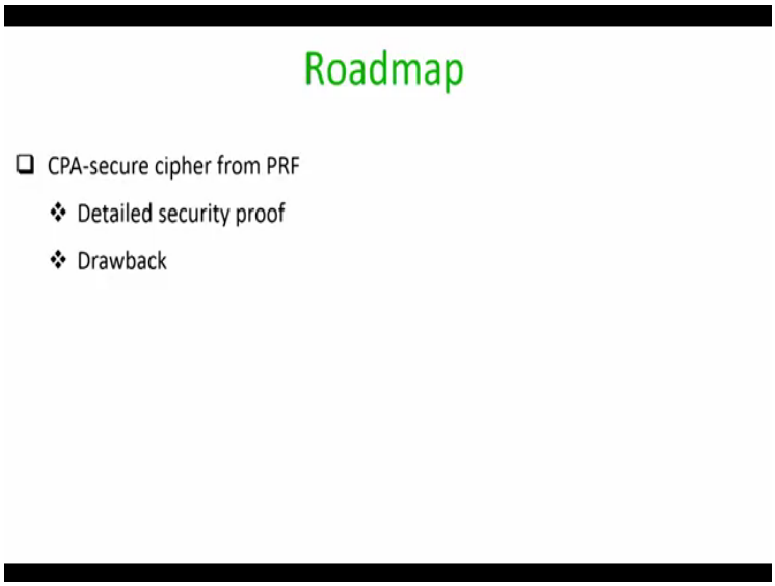
Foundations of Cryptography
Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
International Institute of Information Technology-Bangalore

Lecture-15
CPA Secure Encryption From PRF

Hello everyone, welcome to lecture 14.

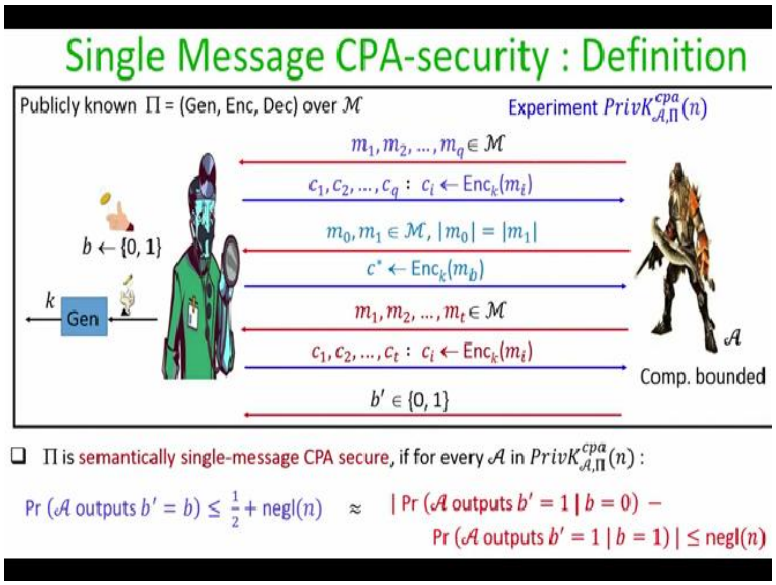
(Refer Slide Time: 00:32)



The plan for this lecture is as follows. In this lecture we will discuss how to design CPA secure cipher using pseudo random function. And we will also see a detailed security proof of this construction, because during this detailed security proof we will see a template of how to prove the security of the constructions based on pseudo random function, which we will repeatedly encounter throughout this course.

We will also discuss some of the drawbacks of the CPA secure cipher that we are going to construct from the pseudo random function, which will motivate us to design what we call as modes of operations of PRF or block ciphers, which we will discuss in the next lecture.

(Refer Slide Time: 01:11)



So, let us first recall the definition of single message CPA security. So, basically this is a game between a challenger and an adversary where adversary is computationally bounded, we have a publicly known scheme over a message space. And the name of the experiment is CPA single message indistinguishability experiment. And basically the experiment consists of 4 phases, you have a pre-challenge training phase, then the challenge phase, post challenge training phase and output phase.

So, during the pre-challenge training phase, our adversary is allowed to adaptively get encryption oracle service that means it can ask for encryption of whatever messages of its choice from the plain text space. And it can ask its queries adaptively and to respond to the adversary's encryption oracle query, the challenger generates a uniformly random key by running the key generation algorithm and encrypts all those plain text which adversary has queried for using the unknown key k, then in the challenge phase, adversary submits a pair of challenge plain text, but the only restriction being that the length has to be the same.

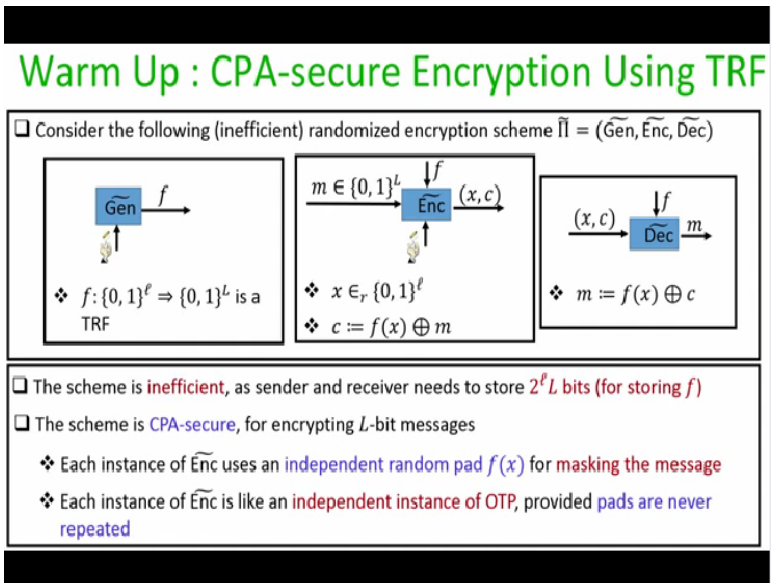
I repeat here that there is absolutely no restriction on the challenge message m_0 and m_1 which the adversary is submitting. It could be any of the messages for which it might have already got the encryption oracle query service and to differentiate the pair of challenge plain text from the encryption oracle queries I am using different fonts for denoting the different plain text. Now to respond to the pair of challenge plain text the challenger basically randomly decides either to

encrypt the message m_0 or m_1 and encrypts some message m_b and the goal of the adversity or the challenge for the adversary is to identify whether c* is an encryption of m_0 or m_1.

And we actually give the adversary more power namely, after seeing the challenge ciphertext c*, we give the adversary access to the post challenge encryption oracle service where again it can adaptively ask for encryption of any message of its choice. And the challenger has to encrypt all those messages again using the unknown key k. And then finally, in the output phase the adversary decides or outputs whether c* is an encryption of m_0 or m_1.

And we say that the encryption process Π is a single message is CPA secure, if the probability that any computationally bounded adversary can win this game is upper bounded by half plus some negligible function negl(n) in the security parameter n or equivalently the distinguishing advantage of the attacker is upper bounded by a negligible probability. So that was our definition of single message CPA security which we had seen in the last lecture.

(Refer Slide Time: 03:54)



Now what we are going to do is, we are going to now design a candidate encryption process which indeed satisfies this notion of security. So before going into the actual construction using pseudo random function for the moment, assume we have got access to a truly random function. And let us first see the construction of a candidate CPA secure encryption process, assuming we have access to a truly random function.

The reason that we are actually seeing this CPA secure encryption process using truly random function is that later when we will see the actual construction based on PRF, the proof will become simplified. So consider this version of encryption process which I call $\tilde{\Pi}$ consisting of a key generation algorithm encryption process and decryption process. The key generation process for this scheme $\tilde{\Pi}$ is basically it outputs a uniformly random function or a truly random function mapping *l* bit strings to L bit strings.

That is the key generation algorithm. Now, the encryption algorithm for this encryption scheme is as follows it takes up plain text of size L bits. And it takes a truly random function which will be available both at the sender as well as with the receiver, and it will be known only to the sender and the receiver. Now, to encrypt the message m, what the encryption algorithm does is it decides a uniformly random x from the domain of the function f, right.

And then truly random function f is evaluated on this x input to obtain a mask of length L bits which is actually used to mask message m, which is the cipher text. Now, you can see that the actual ciphertext is a collection of 2 components. It consists of 2 components namely the x value at which the truly random function is evaluated. And the actual encryption of the message which is the masking of the message with the value of the truly random function evaluated at this uniformly random value x, which is a part of the ciphertext.

How the decryption is going to happen. So, imagine the receiver also having the same truly random function, and it obtains a ciphertext which consists of 2 components the x part and the actual ciphertext part. Now if you see the syntax of the encryption process by the property of XOR, it follows that to recover the message m and what receiver basically has to do is to compute the value of the truly random function f at the input x, and just XOR it back from the ciphertext component of the actual ciphertext or the c component of the actual ciphertext. And that will give back the actual plain text m. So that is the decryption algorithm.

Now why this scheme is inefficient. The reason the scheme is inefficient is that both sender and receiver have to store the description of the truly random function f. And if you imagine a truly

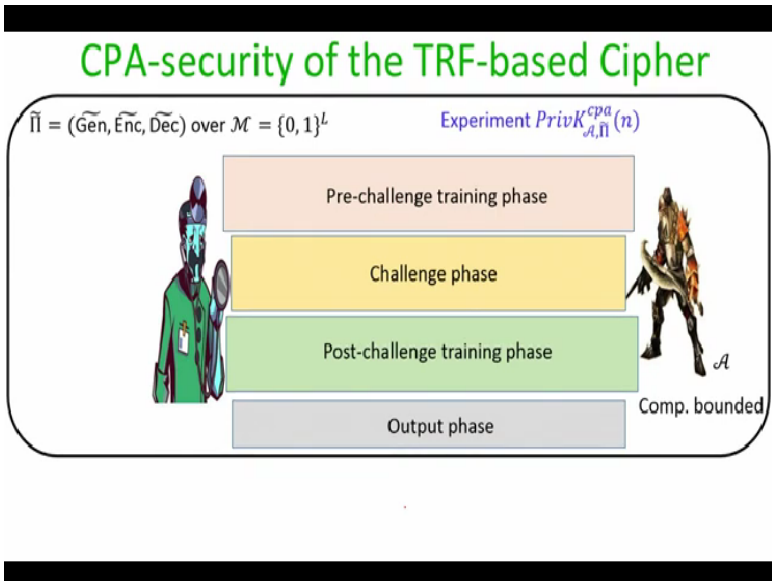
random function as a truth table, then basically both sender and receiver have to store these many number of bits. Namely 2^l rows, each consisting of L bits. So if L is some polynomial function of the security parameter this is exponentially large. So storage wise both sender and the receiver have to store or perform exponential amount of computation. For the moment just do not worry about this inefficiency part, let us analyze the security aspect of this encryption process. We are going to now prove that the scheme is CPA secure for encrypting L bit messages.

And the basic intuition that why this encryption process is CPA secure is that if you see the syntax of your encryption process here, each instance of the encryption process actually evaluates the truly random function on a uniformly random x. That means since that my function f is a truly random function, the value of f on that uniformly random x is going to be a uniformly random L bit string.

That means each instance of an encryption process you can imagine as an instance of one time pad, that this holds under the assumption that the x values which we are actually using for encryption process, never gets repeated. That means if you use this encryption process for encrypting polynomial number of messages, and in those polynomial number of instances of encryption, the x value never gets repeated.

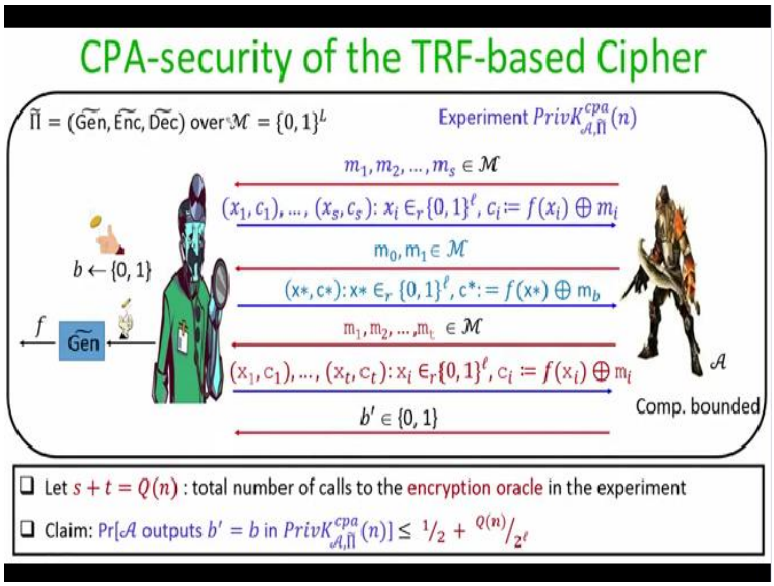
Then each instance of this encryption process you can imagine as an independent instance of one time pad. And we had already seen that if you consider independent instances of one time pad where the pad which is used in each instance is independent of each other, then it cannot be broken even by a computationally unbounded adversary. That means we can say intuitively that conditioned on the event that the x values are never repeated in polynomial number of instances of this encryption process. So, this encryption scheme is actually CPA secure. That is a basic intuition. Now, what we are going to do is we are actually going to formalize this intuition rigorously through our experiment.

(Refer Slide Time: 08:50)



And analyzing that experiment. So, consider any arbitrary adversary A, who wants to participate in an instance of single message CPA security game against this encryption process $\widetilde{\Pi}$ right. So as for the rules of the games, we have 4 phases.

(Refer Slide Time: 09:05)



Then in the pre-challenge training phase, our adversary can ask for encryption of any number of messages of his choice as long as it is bounded by some polynomial function in the security parameter. So let adversary ask for encryption of s number of messages. Now, to respond to these messages, the challenger basically has to encrypt all those messages as per our encryption process $\widetilde{Enc}$.

So for doing that, it is going to run a key generation algorithm or the function generation algorithm, in this case to be specific and generates a truly random function. And then once a truly random function is available with the challenger, it is going to encrypt all the messages for which the adversary has asked for the encryption oracle service, right. So each of the ciphertext (c_i, x_i) basically each ciphertext c_i basically consists of 2 components namely an x component and the actual ciphertext.

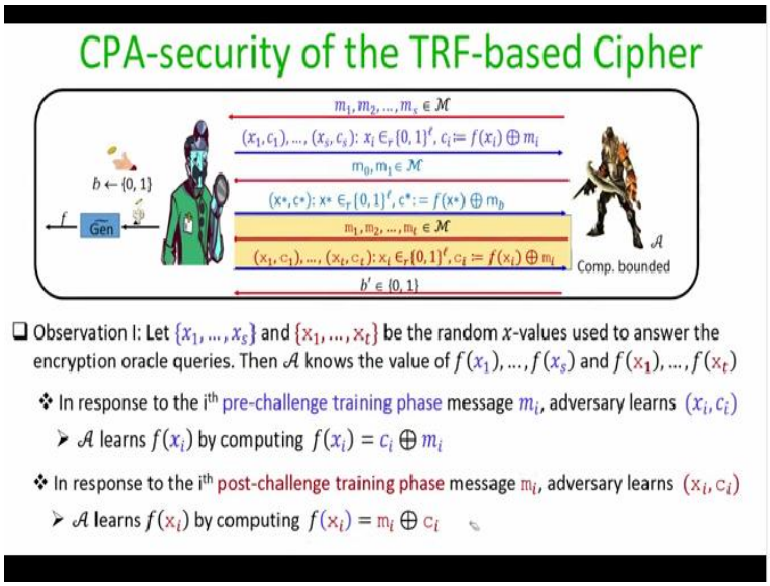
So, the x value is actually used to evaluate the truly random function to obtain the mask and then that is XORed with the actual message to obtain the ciphertext component. In the same way the challenge phase adversary will submit a pair of messages say m_0 and m_1 and our challenger is going to randomly encrypt one of them, I denote the encrypted message encrypted m_b to be the collection of (x*,c*) basically x* is the x value which is used to generate the pad to mask the message m_b. And the masking of the message m_b with the pad is c*, then you have the post challenge training phase we are again our adversary can ask for the encryption oracle service for polynomial number of messages. And again, using the same unknown truly random function which is available only with the challenger, the Challenger is going to generate the mask namely $f(x_i)$ where x_i will be the part of the ciphertext.

And once for x_i is computed it will be XORed with the messages to obtain the corresponding ciphertext part. And then we have the output phase right where adversary basically output whether c*, or the challenge ciphertext (x*, c*) to be more specific is an encryption of m_0 or m_1, right. So let us first do some computations here. Let me denote the total number of queries, namely the pre-challenge queries and the post challenge queries which adversary has asked for is Q(n), which is going to be some polynomial function of the security parameter.

And we are going to prove this claim namely, we are going to prove that any arbitrary adversary A which is computationally bounded and makes Q(n) number of queries in an instance of single message CPA security game against the encryption process Π' or $\widetilde{\Pi}$ is going to correctly identify whether the challenge ciphertext is an encryption of m_0 or m_1 with probability at most $1/2 + (Q(n) / 2^l)$. Since Q(n) is of some polynomial function of the security parameter i^{th} and assuming

that l is also some polynomial function of the security parameter, overall this quantity $Q(n)/2^l$ is actually a negligible quantity, right. That means we are actually going to prove that the probability that any arbitrary adversary A by Q(n) number of queries wins an instance of single message CPA security game is half plus negl(n), which proves that our encryption process $\tilde{\Pi}$ is actually single message CPA secure.

(Refer Slide Time: 12:34)



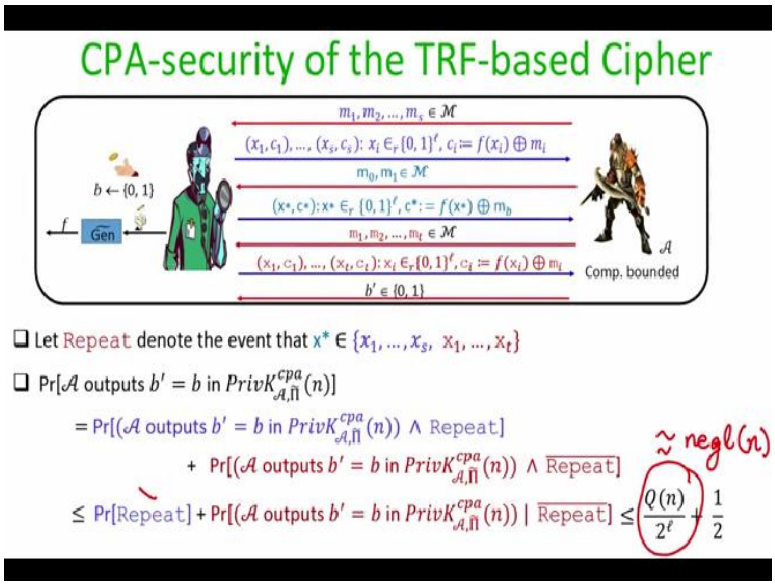
So let us proceed to prove our claim. And we are going to prove it using several observations that we are going to make about any instance of single message CPA security game. So our first observation is that if we consider the set of x values which are actually used for encrypting the a pre-challenge encryption or queries, and x values which are actually used by the challenger to encrypt the post challenge encryption oracle queries, then adversary knows the value of the truly random function f available with the challenger on those x values, why?. So, let us first focus on the pre-challenge training phase. So, if I consider that i^{th} query right so, i^{th} query is for the encryption of the message m_i where m_i is known to the adversary. And in response to that, adversary learns the ciphertext which is collection of (x_i, c_i) value and adversary knows the relationship between m_i x_i c_i.

Namely adversary knows that the unknown truly random function f evaluated at the known x_i is related to c_i and m_i by the XOR operation and everything is known to the adversary in this equation except the f part. So, if adversary knows the m_i part the adversary knows the c_i part and

adversary knows the x_i part. So, if adversary just performed this operation, adversary comes to know about the value of the truly random function f at the x_i.

So that is the case for the pre-challenge query phase or pre-challenge training phase. And the same holds even for the post challenge training phase. That means if adversary has asked for the encryption oracle service for this message m_i and in response to that adversary obtains the ciphertext denoted as (x_i, c_i) then that by performing this operation, namely performing the XOR of m_i and c_i adversary will learn the value of the truly random function f at the input x_i right. So that is the first observation which is easy to see.

(Refer Slide Time: 14:32)



The second observation here is if you consider x* value namely the x value, which is actually used to generate the pad for encrypting the challenge, which is actually used to generate the challenge ciphertext right, then if this x* does not belong to the set of x values, which are used to encrypt a pre-challenge query phase, pre-challenge queries and the post challenge queries, then the probability that adversity wins this instance of CPA game is at most 1/2.

And this is because of the fact that if the x* value which is actually used to generate the challenge ciphertext has never been used in any of the pre-challenge training phase query or the post challenge phase query, then basically, if I shave off this pre-challenge training phase and the

post challenge training phase, and if I just focus on the challenge phase of this experiment and it reduces to an instance of one time pad indistinguishability experiment.

And we had already seen that one time pad is perfectly secure even against a computationally unbounded adversary. So that is proof that proves our second observation right. Now, the third observation that we can make about this instance of CPA game is that, if the x* value which is used to generate the challenge ciphertext is repeated that means it belongs to either the set of x values which are use during the pre-challenge training phase, or the set of x values which are used during the post challenge training phase, then the probability that adversary wins this game is actually 1. This is because as part of the challenge ciphertext adversary will know the x value, namely the x* value, which is used to generate or encrypt the message m_b. And adversary has also seen the x values used during the pre-challenge training phase, as well as post challenge training phase. So by just comparing the x* value with the list of x values that the adversary has, if it sees that x* is getting repeated, then it can easily identify where the c* is actually an encryption of m_0, or whether it is an encryption of m_1 by performing this operation, right. So that is our third observation. And the most crucial observation that we are now going to make is that the probability that the x value which is used in the challenge ciphertext namely x* gets repeated in when our adversary is make Q(n) number of queries is at most $(Q(n)/2^l)$ right.

So for this, notice that Q(n) that is nothing but the number of queries that the adversary has made throughout this instance of the CPA game. And in each instance of the query when adversary is asking for the encryption of some message, the x value which is used by the challenger to encrypt that message is randomly picked over the set of l bit binary string, right. So the probability that x* which is actually used in the challenge, to generate the challenge ciphertext gets repeated in Q(n) number of queries is of course $Q(n)/2^l$.

So that is our fourth observation right. So based on these 4 observations, let us formalize our argument as to why this truly random function based scheme is indeed CPA secure. So let Repeat denote the event that the x value which is used to encrypt the challenge, which is used to generate the challenge ciphertext gets repeated. That means it is either used during one of the pre-challenge encryption oracle query or one of the post challenge encryption oracle query.

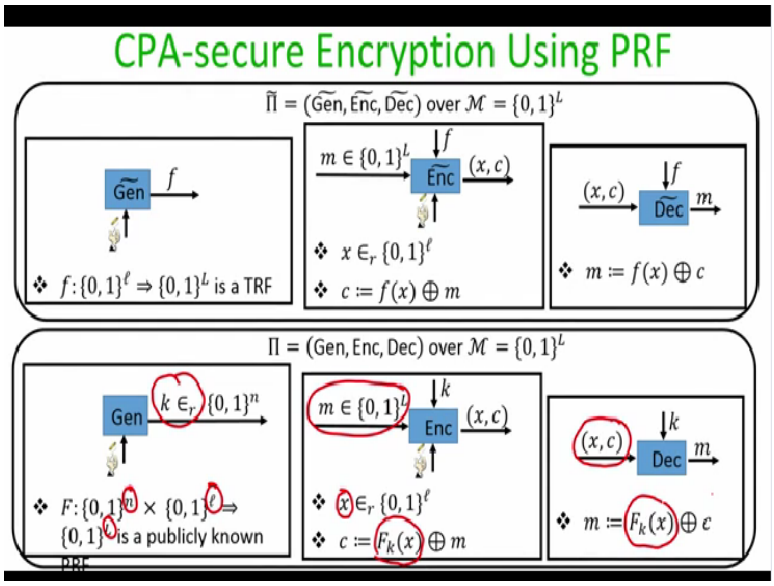
Then our goal is to analyze what is the success probability of our adversary of identifying correctly whether the challenge ciphertext is an encryption of m_0 or m_1. And the winning probability can be split into the sum of 2 individual probabilities, namely, the probability of adversary winning the CPA game given that the x* value gets repeated during any of the encryption oracle queries, or the probability that adversary wins CPA game, given that the x* values never gets repeated during any of the encryption oracle queries, right.

Now, what I can do is that the first probability here, I can always upper bound by the probability that the event Repeat occurs and the second probability I am just writing it down as it is. So that means my goal is to now calculate the probability that the event Repeat occurs and the probability that adversary wins the CPA game, given that the event Repeat does not occur. And I am going to show that this is upper bounded by $1/2 + Q(n)/2^l$.

Why well we have already seen that the probability that the x* value gets repeated is $Q(n)/2^l$, we had already seen in one of the observations, and the second probability, namely adversary wins the CPA game given that x* value never gets repeated is upper bounded by 1/2 because in this case, we can simply say that the pre-challenge training phase and the post challenge training phase is useless for the attacker.

And if I just focus on the challenge part of the game that is nothing but an instance of one time pad indistinguishability experiment, and we know that even a computationally unbounded adversary cannot win the one time pad indistinguishability game. So that is how we get the overall success probability of our adversary right. And as I already argued that the probability $Q(n)/2^l$ can be approximated by a negligible function in the security parameter, if Q(n) is a polynomial function in the security parameter and l is also polynomial function in the security parameter. So what we overall get is that a truly random function based construction is indeed CPA secure.

(Refer Slide Time: 20:28)



But as I said, that hypothetical scheme $\widetilde{\Pi}$ based on the truly random function we cannot use in practice because for deploying it both sender and the receiver have to store the description of a truly random function which is exponentially larger in size. So, what we are going to instead do is that we are going to retain the blueprint that we had used for this encryption process $\widetilde{\Pi}$. And what we are now going to do is that wherever there were invocations of the truly random function, we replace them by invocations of a pseudo random function.

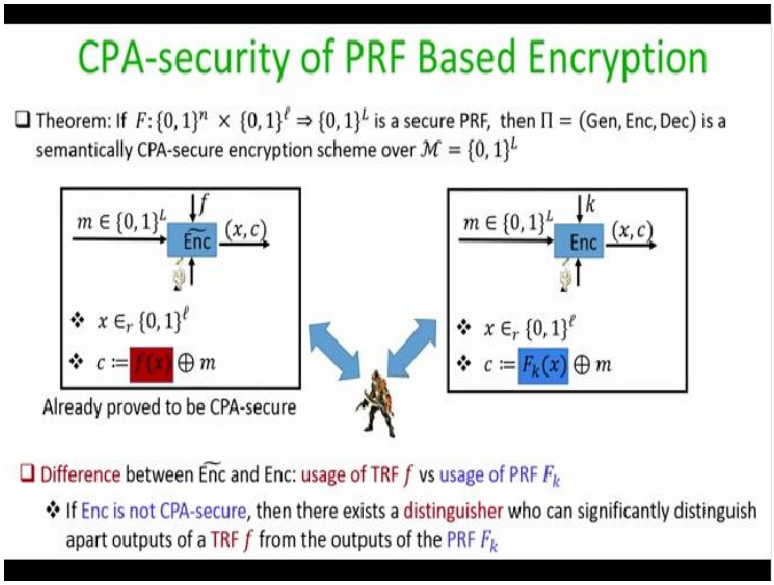
So here I assume that I have a pseudo random function F, whose key size is n bits, block size is *l* bits and output size is L bit, right. So instead of outputting a truly random function, the modified key generation algorithm is now going to output a uniformly random key, which will be available both with the sender and the receiver. In the same way, instead of computing the value of truly random function on some x input to generate the mask, and using it to masking the message, what we are going to do is if the sender wants to encrypt a message of size, L bits, it is going to uniformly random choose an x value, evaluate the keyed pseudo random function on that x input to generate a pad of size L bit and XOR it with the message. So operation wise we are more or less doing the same process as we are doing in the truly random function based encryption process.

The only difference is that instead of evaluating a truly random function on some random x value we are now actually evaluating a keyed pseudo random function with respect to a uniformly

random key on some uniformly random x value. And the modified decryption operation process is analogously similar to the truly random function based decryption process.

Namely if the receiver sees a ciphertext, it passes it as a collection of x part and actual ciphertext part. The x part is used to evaluate the keyed function with the same key k available with the receiver to generate the mask and this XOR back from the ciphertext to recover back the plaintext. That is a modified scheme pi.

(Refer Slide Time: 22:46)



And a theorem statement that we are now going to prove is that if the function F is a secure pseudo random function, then the modified scheme is actually a CPA secure encryption process for encrypting L bit messages. And before going into the actual proof, let us compare the 2 encryption process, on your left hand side part you have the truly random function based encryption process, and in your right hand side part, you have the pseudo random function based encryption process. And what is the actual difference between these 2 encryption process, the only difference is the nature of the mask. In the truly random function based encryption process, the mask was generated by evaluating an unknown truly random function where the function was known only to the sender and the receiver.

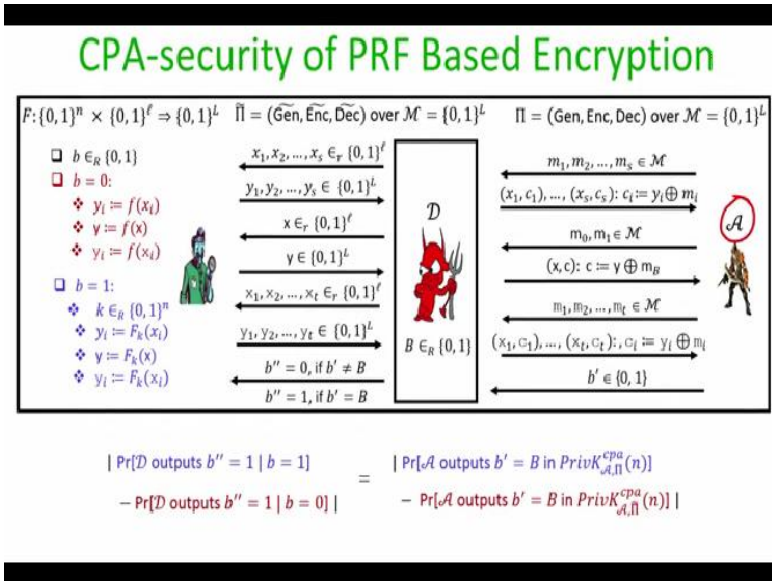
Whereas in the PRF based construction, the mask is generated by evaluating a keyed pseudo random function where the key is now random and known only to the sender and the receiver,

right, that is the only difference. And we had already rigorously proved that truly random function based encryption process is CPA secure against any computationally bounded adversary.

So intuitively the same should hold even if I replace all the instantiations of the truly random function by a pseudo random function. Because now if after the replacement of truly random function and by a pseudo random function the modified scheme is not CPA secure, that means we have now a distinguisher, who can distinguish apart the behavior of a truly random function f from the behavior of a pseudo random function F(k).

But that is a contradiction to the assumption that we are making about the function F being a secure pseudo random function. So now we are going to rigorously formalize this intuition through a reduction.

(Refer Slide Time: 24:34)



So now you have 3 entities here you have the truly random function based encryption process, which we know is CPA secure. We have the modified scheme based on pseudo random function. And both these schemes are over the same message space of L bit strings. And we have a publicly known pseudo random function. Now imagine we have an adversary A who can actually win the CPA security game against a modified scheme Π.

Now using this adversary A, what we are going to do is we are going to design a distinguisher, a PRF distinguisher, who can distinguish apart the output of a pseudo random function from the output of my keyed function F, which will be a contradiction. So let us see how the distinguisher operates. So distinguisher basically acts as a verifier and invokes the adversary A the CPA adversary A. And an instance of the CPA game is run or executed.

So as per the rules of the game adversary asks for the encryption queries, namely, it asks for the encryption oracle service and it throws a s number of queries. Now to encrypt these messages, what the distinguisher does is it randomly chooses s number of x values, right from the domain of either the keyed function F or the truly random function f, and it asks for the function values at is x values and in response, it gets the function values of output of the function on those x values.

So this part of the experiment is basically the PRF game right, where the adversaries goal is to basically query for several number of x value of the function and gets the response and based on the response, he has to find out whether he has interacted with a truly random function or a pseudo random function, whereas this part of the game is your CPA game right. That is the way you can pictorially imagine the reduction right.

So once the distinguisher sees the values of the x inputs and obtain the y values here is how it produces the ciphertext and forwards to our adversary in the CPA part of the experiment. Namely the x value is retained as it is. And the c part basically is the XOR of the messages for which the adversary has asked for the encryption XORed with the y values. Now as per the rules of the CPA game in this part of the experiment adversary is going to submit a pair of challenge plain text.

And what the distinguisher does is the following. It randomly decides either to encrypt the message m_0 or m_1. And to encrypt that message m_b, it picks a uniformly random x value from the domain of the function and asks for the function value added x input and in response, it gets back the y value. And the challenge ciphertext for the CPA part of the experiment is computed as this ciphertext basically consist of the x value used by the distinguisher PRF distinguisher.

And the c part is basically the XOR of the y value with the message m_b where m_b could be either m_0 or m_1. In the same way in the CPA part of the experiment, our adversary will ask for the post challenge training phase, post challenge encryption oracle queries and for responding to that the PRF distinguisher basically picks up several random x values and asks for the function values at those x inputs in response it gets back the y values.

And it submits the ciphertext to the CPA adversary by repeating those x values and the c values being the XOR of the y values and the m values right. Now before going into the details, if I consider the PRF part of the PRF game here, the y values as generated by our PRF challenger would have been generated as follows. The PRF challenger was basically tossed a fair coin, which outputs 0 and 1 with probability 1/2.

And if b would have been 0, then all this y values right used for the responding to this x inputs are actually generated by a truly random function. And if all these y values are generated by a truly random function, then all the ciphertext which are actually forwarded to our CPA adversary will have the same distribution, as in the instance of a CPA game against encryption process by $\tilde{\Pi}$ right.

That is what is the relationship between the probability distribution of the y outputs in the PRF game and the ciphertext in the CPA game, right. On the other hand, if b = 1 in the PRF experiment, then all these y values, right would have the same probability distribution as if they are generated by querying a keyed pseudo random function, which further implies that that the ciphertext with our CPA adversary is seeing will have the same probability distribution, as our CPA adversary would have seen in an instance of the CPA game against the scheme pi.

Notice that neither of our PRF distinguisher D nor our adversary CPA adversary A knows what exactly is the probability distribution of the ciphertext and the y values right because we are using the distinguisher and adversary in a blackbox fashion, their goal is basically to win their respective experiments. But this is the property which holds with respect to the probability distribution of the y values in the PRF game and the ciphertext in the CPA game.

Now, we are going to make some claims here. Right before going into that the last part of the CPA game will be that adversary will output whether this (x, c) is an encryption of m_0 or m_1, namely it outputs a b'. And based on that our distinguisher PRF distinguisher has to output whether the y values that it has seen are the response from a truly random function or the keyed function F(k).

So the way our PRF distinguisher outputs is the following. It checks whether the CPA adversary is correctly able to identify whether (x, c) is an encryption of m_0 or m_1. That is the case then our PRF distinguisher says that the y values are generated as per the pseudo random function. Otherwise it says that the y values are generated by a truly random function. That is the way our distinguisher is designed.

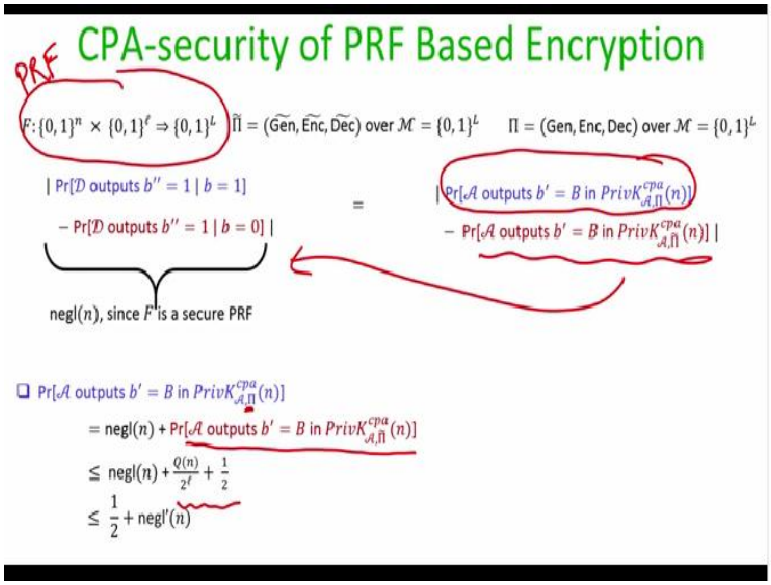
Now, if I find a running time of what PRF distinguisher well, the running time of PRF distinguisher is of the same order as the CPA distinguisher that means if the CPA distinguisher is polynomial time, then so is our PRF distinguisher. Now, let us analyze the property of the so called PRF distinguisher that we have designed. I claim that the probability that our PRF distinguisher outputs b'' equal to 1, given that b = 1 is the same as the probability that a CPA adversary outputs b' = B in the CPA in an instance of the CPA game against the encryption process Π.

This is because as I said earlier, if b = 1 in the PRF game, that means the y values are generated by a pseudo random function, then, right all the ciphertext that the CPA adversary is seeing will have the same probability distribution as per the encryption process Π. So with whatever probability of a CPA adversary will be able to correctly identify (x, c) being encryption of m_0 or m_1 with exactly the same probability our PRF distinguisher was going to output b = 1 right.

That is a first claim here. And in the same way, as I explained earlier, that if b = 0 in the PRF game, that means all the y values generated by querying a truly random function, then the probability that our PRF distinguisher still incorrectly labels them as the output of pseudo random function is that the probability that of a CPA adversary wins the CPA game, against instance of truly random function based encryption process.

And this is because the decision for the PRF distinguisher is that it outputs b = 0 only if the CPA adversary correctly identifies whether (x, c) is an encryption of m_0 or m_1 right. That means in summary, what I can say is that if I consider the distinguishing advantage of our PRF distinguisher, then that is exactly the same as the distinguishing advantage of our CPA adversary, right. That means with whatever probability it can win the CPA game against a new sense of the scheme Π minus the probability that it can win the instance of CPA game against a modified scheme Π', right.

(Refer Slide Time: 33:48)



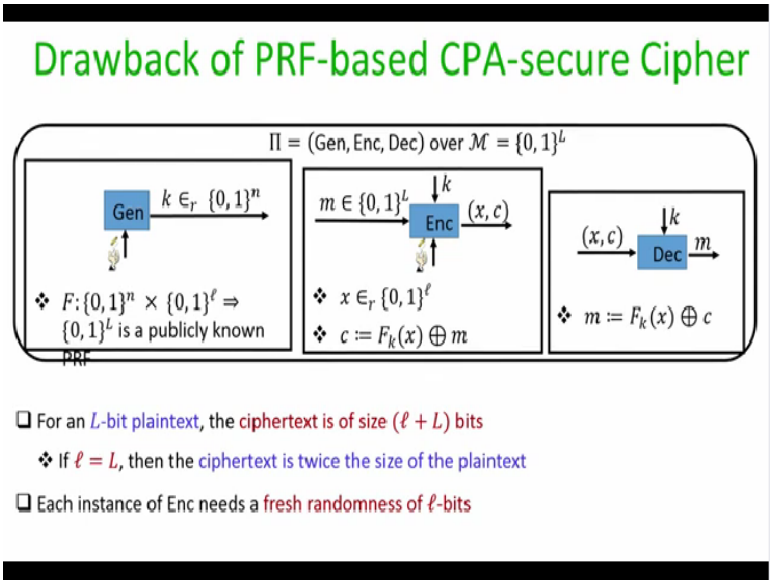
So that is a fact we have already established. But our goal was to basically compute a winning advantage or the probability that our adversary can win the CPA game against an encryption of Π. So if I take or reshuffle the terms, I get the following. So notice that the left hand side namely, the distinguishing advantage of our PRF should be negligible, because as per our assumption, we assume that the function F is actually a PRF.

That means as per the definition of PRF, or a secure PRF, for any polynomial time distinguisher, the distinguishing advantage of the distinguisher should be upper bounded by a negligible function. So, if I take this remaining term to the left hand side, what I obtain is that the probability that our adversary wins the CPA game against the encryption process Π is a summation of these 2 probabilities.

And we already calculated that for any arbitrary adversary A, the probability that it wins an instance of the CPA game against the truly random function based encryption process by making Q(n) number of queries is upper bounded by $1/2 + Q(n)/2^l$ which is same as half plus negligible. So, I can replace this quantity by some another negligible function negl' (n).

So, overall we get at the probability that our adversary A wins the CPA game against PRF based construction is upper bounded by half plus negl'(n), which is satisfying the definition of CPA secure scheme.

(Refer Slide Time: 35:24)



That means we have proved that the scheme Π which we have designed based on pseudo random function is indeed CPA secure. So, we now have a candidate CPA secure scheme, but there are certain limitations of this scheme. The first limitation is that if you have a message of size L bits, then the overall size of the ciphertext actually is *l* plus L bits because remember that the x value which we are using to generate the mask is also has to be a part of the overall ciphertext.

More specifically, if L and *l* are same that means our ciphertext size is twice size of the plaintext right. And that means if I have a large message consisting of several blocks of L bits, and if *l* is same as L then actually the ciphertext is exactly the twice as the size of the plaintext. So that is

the first limitation of this or the first drawback of this scheme, not a limitation. It is a drawback on this scheme.

More importantly, each instance of the encryption process here is going to require a fresh instance of randomness namely the x value which we are using to generate the mask has to be freshly picked for each instance of the encryption process right. So that is the second drawback because in practice generating uniform randomness is a computationally expensive task.

So these are the 2 restrictions or the drawbacks of the candidate PRF based scheme that we have designed here. In the next lecture we are going to see that how we will proceed to get rid of these 2 restrictions. To summarize in this lecture we had seen a candidate encryption process based on pseudo random function, and we had seen a rigorous security proof that how we can reduce the overall security of this constructed scheme to the security of the underlying pseudo random function. Thank you.