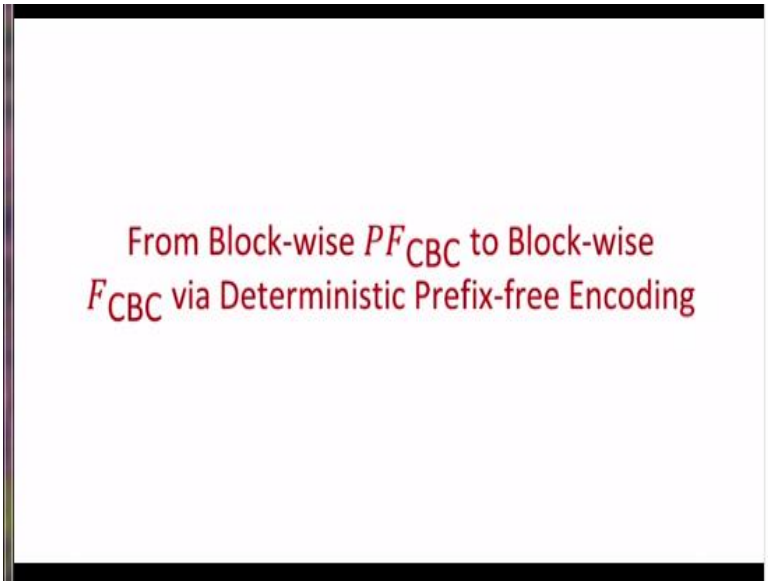
Foundations of Cryptography
Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
Indian Institute of Technology-Bengaluru

Lecture-24
Message Authentication for Long Messages Part II

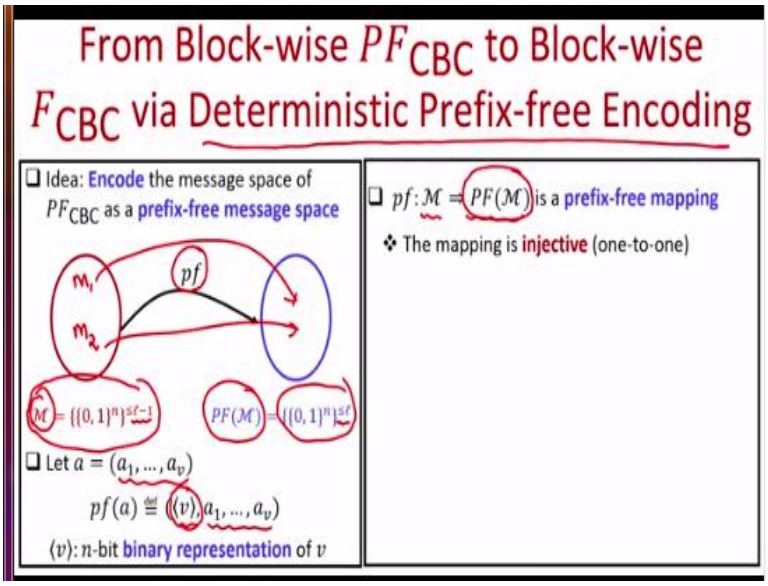
Hello everyone, this is a continuation of the previous lecture and in this lecture we will see how to obtain fully secure block wise fully secure Mac from block wise prefix-free secure MAC. And then eventually we will see how to obtain fully secure MACs for arbitrary long bit strings.

(Refer Slide Time: 00:49)



So now let us discuss the other approaches how to convert a block wise prefix-free CBC PRF to a block fully secure block wise a PRF right.

(Refer Slide Time: 00:59)



And this will be through some encoding and the basic idea here is that what we do is we apply some publicly known encoding function to the message space of the block wise prefix-free PRF. So that the encoded message space constitutes now a prefix-free message space, and what basically it means is that. Imagine you have a message space for the prefix-free PRF consisting of up to l blocks or each of size and l - 1 blocks each of size n bits, what we will do is.

We will do, we will apply some mapping or encoding function which I denote as PF and a encoded message space will now consist of a sequence of block up to little l blocks each of size n bits. And the way a mapping is done here is that is as follows. So, right now we are actually considering a deterministic encoding mechanism . Later on we will consider a randomized encoding mechanism as well.

So, the way this deterministic encoding mechanism works is as follows. So all the input blocks **say** so say u have a sequence of input blocks a, so the corresponding encoded input a will be as follows. So all the message blocks of a are copied as it is in the encoded output and apart from that we now introduce an additional block here, namely the binary representation of the number of blocks in your input a.

So that is the notation this, within the angle brackets you have the binary representation b right. So that is why you can see that the input space could consist of a 2 l - 1 blocks. And since we are

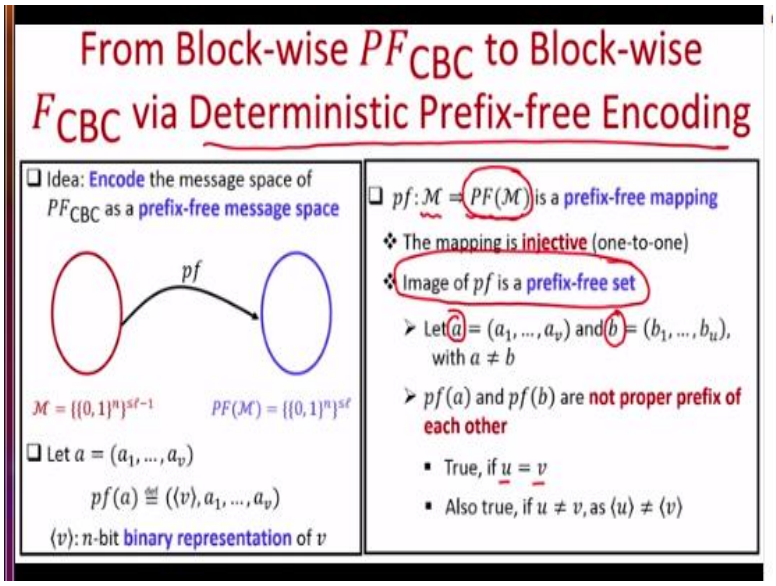
now introducing an additional blocks in the encoded output, that is why the outputs now can consist of up to little l blocks. We have an additional block here introduced because of the encoding process.

And the idea here is that using this deterministic encoding process. We hope that instead of operating the prefix free CBC PRF over the message space. We are now going to operate it over the encoded message space. And since the encoded message space will constitute a prefix preset, our distinguisher cannot in issue queries which will constitute a prefix preset and hence the overall construction will be a fully secured PRF.

So now let us prove that the deterministic encoding which we have seen here indeed constitutes a prefix free mapping. That means the mapping from the message space, which could be a non prefix free set to the encoded set gives you an encoded set which actually constitutes a prefix free set that is what we want to prove. So first of all it is easy to see that a mapping is injective unable one to one that means if you have 2 inputs, m 1 and m 2 consisting of sequence of blocks they up to a little l - 1 blocks.

And if we encode them then they are going to be different if m 1 and m 2 are different, because if m 1 and m 2 are different. Then definitely the blocks which are present in m 1 and blocks which are present in m 2 will also carry over in the encoded m 1 and in the encoded m 2 and they will be differing and hence the encoded m 1 and encoded m 2 will be different. So that proves that your mapping that we have defined here is indeed a 1 to 1 mapping right.

(Refer Slide Time: 04:25)



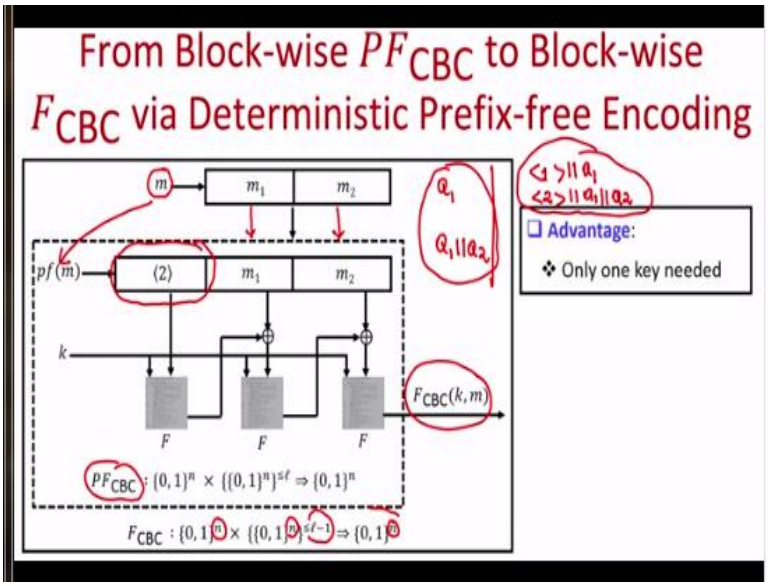
Now we want to prove that the image set of PF namely all the encoded messages that we get as per this mapping constitutes a prefix preset. So imagine you have 2 inputs belonging to the message space say the input sequence a and another input sequence b consisting of v and u number of blocks and say a and b are different inputs, right. So my claim is that the encoded input a and a encoded b input b are not proper prefix of each other.

And if you put prove this for an arbitrary input a and a arbitrary input b that proves that the encoded proves that the mapping that we have considered indeed gives you a prefix preset. So clearly the claim that we are making about encoded a and encoded b is true if the number of blocks in a and a number of blocks in b are same namely if u and v are same. But if a and b are different then definitely at least one of the blocks in the encoded a and encoded b will be different.

Because the all the blocks in a and all the blocks in b cannot be same even if a and b are different. So the claim is trivially true if u = b u = v, whereas if u is not equal to v that means if the number of blocks in a and number of blocks in b are not same, and even and still if a is not equal to v **a** a is not equal to b. Then definitely the binary representation of the number of blocks in a and the binary representation of the number of blocks in b will be different.

That means the first block which we are actually putting in the encoded a and the first block which we are putting in the encoded b will be different, which will ensure that overall the encoded a and encoded b are different. So that proves our claim that the image set of the mapping that we have considered here indeed gives you a prefix preset right.

(Refer Slide Time: 06:21)



So, now let us see how we apply this prefix-free encoding and to a prefix-free secure CBC block wise secure PRF and get a fully secure block wise PRF right. So our goal is to construct a block wise fully secure PRF taking and bit key and a sequence of blocks as input up to l - 1 blocks each of size little n bits. And to obtain a fixed size output of size little n bits and for demonstration assume that we want to operate this block wise fully secure **c** CBC PRF on an input consisting of set 2 blocks m 1 and m 2.

So what we will do is that we will first convert this input m by applying our deterministic prefix-free encoding namely this message m is converted into an encoded input. And when we converted into in encoded input then the message blocks are repeated as it is and apart from that at the beginning we now have a new block namely the binary representation of the number of blocks that are there in right.

So now we have one more block compared to the number of blocks which are present in m and once we have the encoded input what we do is we apply the existing construction namely the

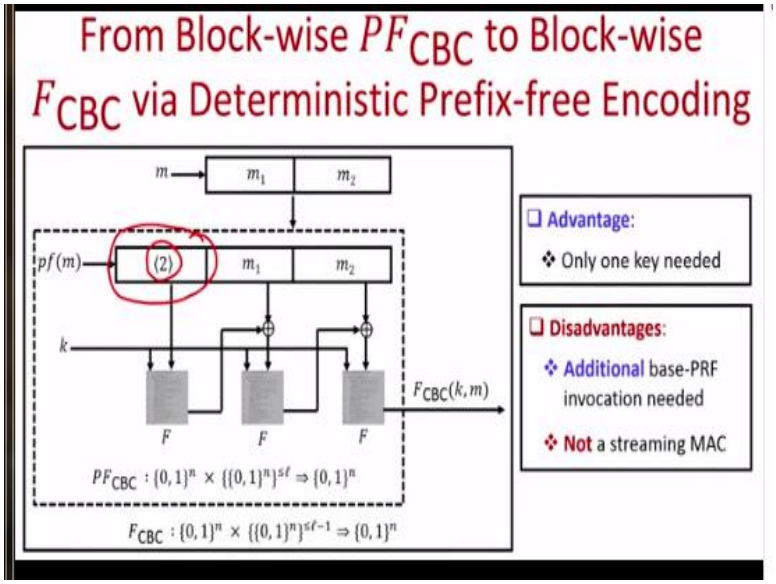
block wise prefix-free secure CBC with the key of size n bits. And the output will be considered as the output of the block wise fully secure CBC for the message m under the key k right, now why this construction is intuitively secure.

Because now even if an adversary is asking, even if their adversary ask for the function output on a input a 1 and on an input a 1, a 2, actually, he will be getting the output of the function on the input sequence a 1 concatenated with a 1 and the output sequence sorry, he will get the output of the function on this input, and he will get an output on this sequence right.

And hence, he cannot launch the attack that we had seen when we discuss the insecurity of the block wise prefix-free secure PRF. Because even now, he is making queries which constitute a prefix preset, basically those queries are converted it to another set of queries, which is not a prefix preset and from the viewpoint of an adversary is now as good as he is interacting with a normal PRF.

So that is how overall idea of the security proof but the formal details are indeed advanced in nature and that is why I am skipping the details here. So if you see the advantage of this construction, namely this way of constructing block wise fully secure PRF via deterministic encoding.

(Refer Slide Time: 09:13)

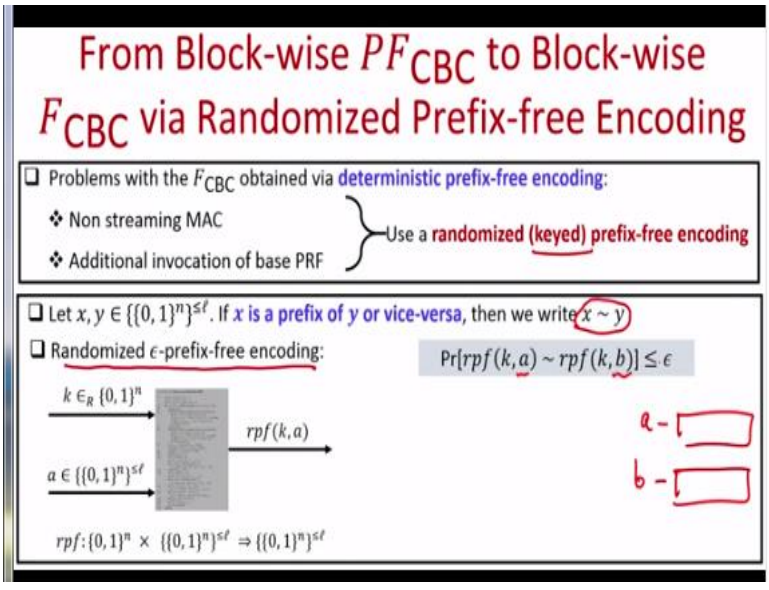


Then the advantage is that we need only one key unlike the encrypted CBC PRF, where we need to operate with 2 keys. However, the disadvantages are as follows right, so we need an additional PRF invocation right. For this additional block which we are introducing here, ideally we expect a PRF, where the number of base PRF invocations are same as the number of blocks in the message.

But actually in this overall construction, we need to have one additional invocation of the base PRF. And bad news about this construction is that it no longer constitutes a streaming MAC. Because the length of the message or the number of blocks in the message need to be known in advance to the sender. Because if the number of blocks in the message is not known to the sender in advance that then the encoding of the message cannot happen.

Because the encoding requires the binary representation of the number of blocks in the message that is present which has to be inserted at the beginning. But that is why it does not gives you a streaming MAC right. So that is the approach of going from block wise prefix free secure PRF to a block wise fully secure PRF using deterministic encoding.

(Refer Slide Time: 10:30)



And now we see how we can go from this block wise prefix free secure PRF to block wise fully secured PRF using a randomized encoding a randomized prefix free-encoding. So the reason we want to go for a randomized prefix free-encoding is that the problems that we face with the

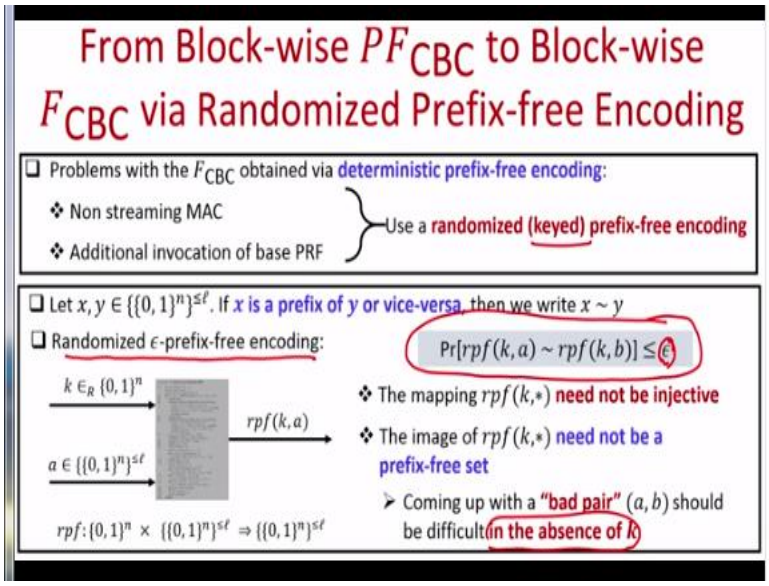
deterministic prefix free-encoding is that we do not get a streaming MAC. And we need to have an additional invocation of the base PRF.

And the solution to get around these 2 problems is to use a randomized prefix free-encoding which will be a keyed encoding and the way we do the encoding is as follows. So the first introduce some notations here. So imagine you have 2 block sequence input say input x and input y each consisting of up to l blocks each of size n bits and if x is a prefix of y or if y is a prefix of x, then we used a notation x is related to y by this symbol right.

So now we introduced the definition of what we call as randomized epsilon prefix free-encoding, where epsilon is some parameter and what this encoding does is. It takes as input a key of size a little n bits and a block sequence of consisting of up to little l blocks and it gives you an encoded output which is a keyed encoded output for the input a. And when I say that it is a randomized epsilon free-encoding, what it means is that the probability to obtain probability that you obtain 2 inputs a and b from the domain input domain consisting of up 2 little l blocks.

Such that under the same key k, the corresponding encoded outputs are related to each other by this prefix relation is upper bounded by epsilon. That means if you have one sequence a and if you have another sequence b then the chance that encoded a and encoded b is prefix of each other or the encoded a is a prefix of encoded b or the encoded b is a prefix of encoded a is upper bounded by epsilon. That is what I mean when I say that I have a randomized epsilon 3 encoding.

(Refer Slide Time: 12:44)



So basically the idea here is that we want to the idea here is that if there is an adversary or computationally bounded adversary, and if it does not know the value of the key k that which we are actually doing the encoding. Then for the adversary it is very difficult to come up with a pair of inputs a comma b consisting of up to little l blocks, such that either the encoded a is a prefix of encoded b or the encoded b is a prefix of encoded a.

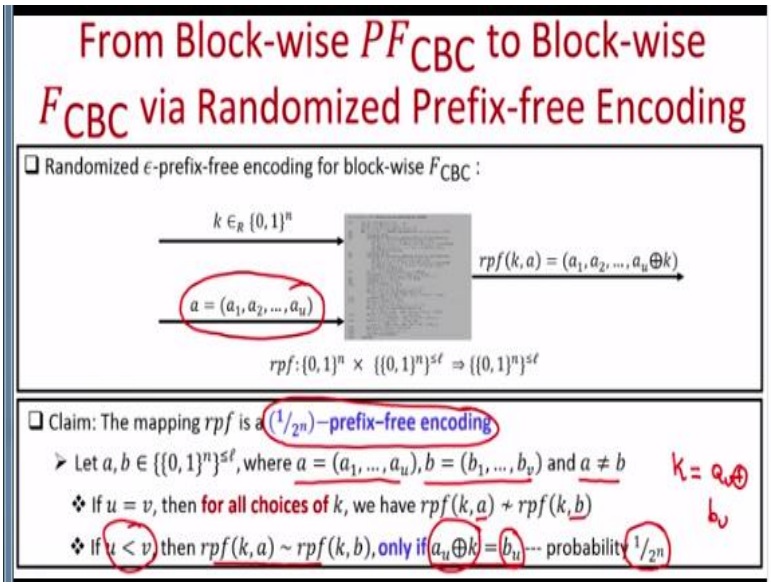
That is what is the security requirement from this randomized epsilon free-encoding. I stress here that the mapping or the randomized the way this randomized encoding is operated. It is not necessary that the mapping should be an injective mapping that means you can have multiple is a's whose encoding will be the same under the key k.

But the security guarantee that we obtain from this epsilon prefix free-encoding is that even though there could be several such candidate is which under the key k would have given you the same encoded output. The probability of finding such multiple a's in polynomial time should be upper bounded by some negligible quantity or by the quantity epsilon. I also stress that unlike the deterministic prefix free-encoding.

The encoding that we are constructing here namely the keyed encoding it may not give you a prefix free set. However even if it is not going to give you a prefix free set the chance that adversary could come up with bad pair of input a comma b. Such that encoded a is a prefix of

encoded b or vice versa should be upper bounded by the probability epsilon and if you ensure that epsilon is very, very small, then we are done right.

(Refer Slide Time: 14:30)



So now assume for the moment you have ok, so what we are going to now do is we are going to define a randomized epsilon prefix free-encoding for our CBC PRF right. And the way we define this encoding is as follows, so imagine you have an input consisting of several blocks a u number of blocks, where u is upper bounded by little l and each block is of size n bits and we have a k key with which we want to be encoding right.

So the encoded output a will be as follows, you repeat all the blocks of a except the last block. And a last block is basically the XOR of the last block of the actual a and the key k, so that is how we are actually computing the encoded a. And my claim is that if the key for this encoding is selected uniformly randomly from the set of little n bit strings. Then this mapping constitutes a 1 over 2 to the power n prefix free encoding.

That means the probability that an adversary who does not know the value of k can come up with 2 inputs a and b such that either the encoded a is a prefix of encoded b or vice versa is upper bounded by probability 1 over 2 power n, which is definitely a negligible function in the security parameter right. So let us prove that, so imagine you have a block you have an input a, a block sequence a and another input b, a sequence of blocks and say a and b are different right.

And we want to find out what are the chances that the encoded a and encoded b either the encoded a is a prefix of encoded b or vice versa. So first of all notice that if the number of blocks in a and a number of blocks in b are same. And if a is not equal to b then for every value of the key definitely neither the encoded a will be a prefix of encoded b or vice versa because when we do the encoding right all the blocks of a except the last block and all the blocks of v except the last block will be present in the respected encoded outputs.

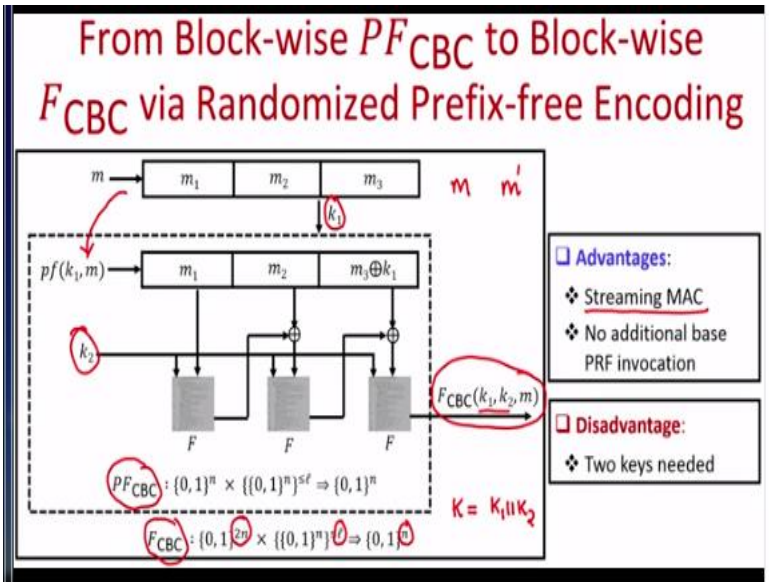
And a last block will be the XOR with respect to the key, so definitely all of them will be different neither will be prefix of each other. On the other hand consider a scenario, where say without loss of generality that the number of blocks in a is less than the number of blocks in b and a is not equal to b. Then what is the probability that encoded a is a prefix of encoded b, well encoded a will be a prefix of encoded b, if and only if the following relationship holds.

Namely, the last block of the encoded a will be this, the XOR of the key with the last block of a, whereas the uth block right. I am assuming here is u is less than v. So, the uth block of the encoded b will be b u right, there would not be any operation with respect to k. Because there are many more number of blocks in the input b which are still available, because we are in the case where u is less than v.

So only if this relation namely the XOR of a u and k and b u if they are saying we can hope that the encoded a will be a prefix of encoded b. But this holds only if the value of key which is selected for doing the encoding is actually the XOR of the last block of a and the uth block of b. And since the key for the encoding is chosen randomly from the set of little n bit strings, the probability that indeed key is satisfies this relationship is 1 over 2 power n.

That means, the same coding that we have seen is a very simple encoding, but it has a very powerful property that it gives you a 1 over 2 to the power n prefix free-encoding. Namely an adversary a computationally bounded adversary, who does not know the value of the random key k, it cannot come up with a bad pair of inputs a and b such that either the encoded a is a prefix of encoded b or vice versa.

(Refer Slide Time: 18:33)



So now let us see that how by using this randomized prefix free-encoding, we can convert block wise prefix free secure PRF to a block wise, fully secure PRF. And the idea is the same instead of as we used for the case of deterministic encoding right. The only difference is that instead of applying the deterministic encoding, we are going to apply the randomized encoding. So now we will be operating with 2 keys each of size little n bits.

So that is why the overall key for the full block wise fully secure CBC will be of size 2 n bits and it can support a sequence of blocks consisting of up to little l blocks, where each block is of size little n bits and it will give you a fixed size output. For demonstration assume that we have message consisting of up to 3 blocks. So what we do is the overall key for the CBC PRF is interpreted as 2 chunks of little n bit keys.

So the first part is the notice as k 1 and with k 1 we first do the encoding of the input, namely the message m is encoded as per the key k 1 via the randomized prefix free-encoded. And the size of the encoded output and the size of the input remains the same that is important here right. This is in contrast to the deterministic prefix free-encoding, where the encoded output consist of one more block compared to the number of blocks present in the input.

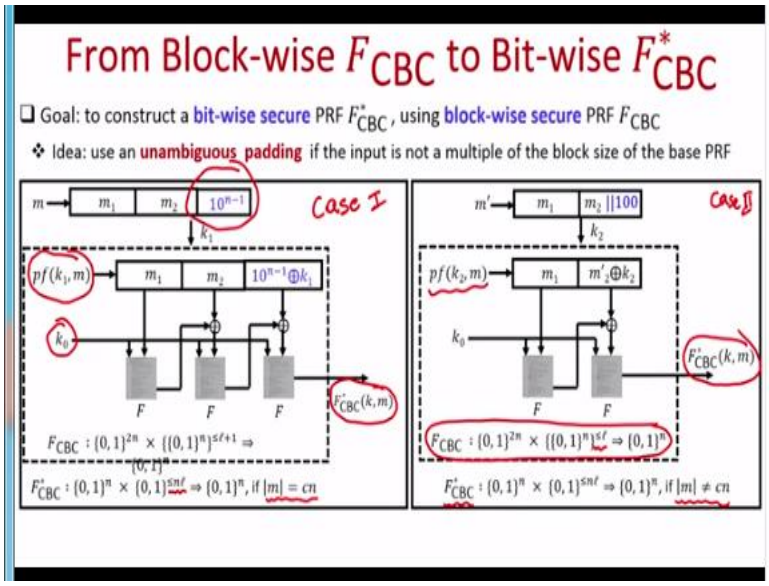
And once we have the encoded input, we now apply the block wise prefix free secure PRF with the second part of the key right. So that is why we need 2 keys one for doing the randomized encoding and one for doing the actual PRF computation. And whatever output we obtained from the block wise, prefix free secure PRF that is considered as the overall outcome of the block wise fully secure PRF under the key k 1 , k 2 for the message sequence m.

Now intuitively, why this construction is secure, because what we have done is with very high probability, we have actually converted the input set of this CBC PRF into a prefix free encoded set right. Even though it is not in principle, a prefix free encoded set, the probability that an adversary without the knowledge of key k 1 would come up with a bad pair of inputs m and m dash, such that the encoded m is a prefix of encoded m dash or vice versa can happen is very, very negligible.

And that ensures that overall construction looks like a normal PRF construction. The advantages of this way of constructing a block wise fully secure PRF is that we do not need the length of the message to be known in advance at the sender right. That is why we obtain now a streaming MAC or a streaming PRF. And we do not need any additional base PRF invocations. So we get rid of both the shortcomings that were there, when we were using a deterministic prefix free-encoding, it was not giving us a streaming MAC.

And we need to have one additional PRF invocation because the deterministic prefix free-encoding requires one additional block in the encoded output, but that is not the case here. However you need to pay a price here, the disadvantage is there here is that you now need to operate with 2 keys, whereas if we it is see the deterministic prefix free-encoding, we were operating only with one key, so now you have a trade off ok.

(Refer Slide Time: 22:06)



So now let us see how we go from constructing bit wise fully secure PRF from block wise fully secure PRF. So remember that our final goal is to construct a PRF or a message authentication code which can take a sequence of bits as input. So, till now the PRF or the MAC that we have constructed could take only blocks or sequence of blocks as inputs. So the idea behind constructing PRF which operates on a sequence of bit is that you first apply some unambiguous padding, depending upon whether the input bit string which you want to feed as an input to your PRF is a multiple of the block size of the base PRF or not.

So for simplicity assume that the block size of the base PRF is little n bits. Now you want to design a PRF which I denote as F star taking a key of size n bits, and it can take any input bit sequence of length up to an l bits right. And now there could be 2 possible cases depending upon whether the input for this PRF F star it is size is equal to some multiple of n or not. So remember I am assuming that a block size of my base PRF is n bits.

So case 1 could be when the input size m is some see constant times n and a case 2 could be when the size of the input is not some is not equal to any constant time same, that means it is not a multiple of n. So depending upon 2 cases, we operate the so called PRF F star that we are interested to construct in 2 different ways. So let us take case 1, in both cases we have to do a padding, so what we do in case 1.

So for instance, imagine my message that I want to input here is of size 2 n bits. That means I can divide it into 2 blocks of n bits. And now I have to do an unambiguous padding, so in this case, actually I do not need to do a padding because my message is already a multiple of n bits. But to indicate it to the receiver that actually I do not need to do a padding, what I am going to do here is I am going to add a dummy block consisting of n bits starting with 1 followed by all 0s.

So that is an indication from the sender side that actually I am not doing any padding. Now to operate F star in this case, what we are going to do is we are going to take a key k 1 which we will soon see how exactly is computed. So remember that the key overall key for F star is just 1 key of size n bits. But what we are going to do is we are going to derive several sub keys and depending upon whether we are in case 1 and case 2, we are going to use some subsets of those sub keys.

So in this case, we use a sub key which I call as k sub 1. And then what we are going to do here is we are going to operate our block wise fully secure CBC PRF with 2 keys each of size n bits. And since it is a block wise fully secure PRF it can take a sequence of blocks of n bits up to l + 1 such blocks. And sorry for the rendering issue here, this is 0 comma 1 rise to power n should be here and it gives you an fixed output.

So, if you see the outer view of the PRF F star that we are going to construct here, it could take an input of size an l bits right. And in this case, if my input already is a multiple of n that means it could have up to l number of blocks. But if you see the inner invocation of the block wise fully secure PRF then it could take up to l + 1 blocks of size little n bits or you might be wondering that why this additional 1 block this additional 1 block might come if actually your message is already a multiple of a.

In which case you need to add a full dummy new block of n bits. So that is why the inner invocation of my block wise fully secure PRF could take up to l + 1 blocks. So internally what I do is I operate, my block wise fully secure PRF and for doing that actually I first take the

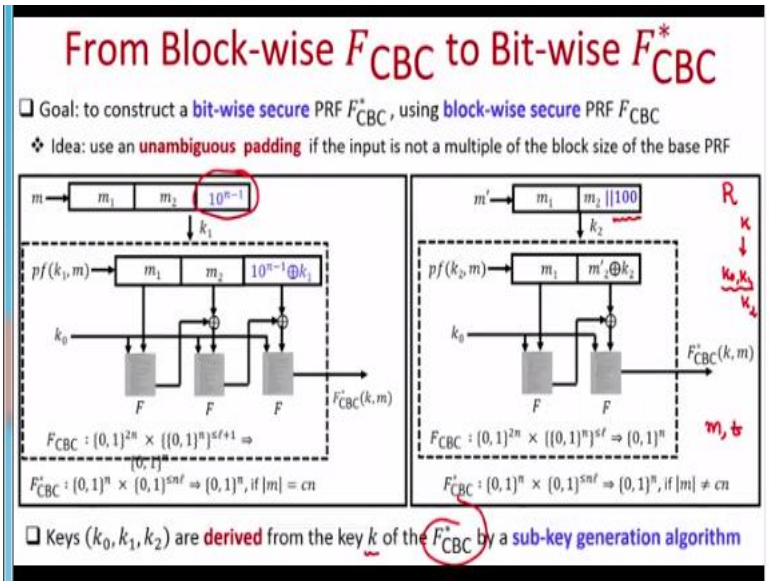
message, the padded message and I apply a prefix free randomized encoding with the sub key k 1 and then I use another sub key k 0.

Again, we will see soon how exactly the sub keys case 1, case 0 are derived. So **we** once we have the prefix free-encoding of the padded message, we take the sub key k 0 and then we run our block wise fully secure CBC and whatever comes out as the output that is taken as the overall outcome of the bit wise secure PRF F star under the key K for the message m. This is case 1 when message length is already a multiple of the block size of the base PRF.

Let us see how the things are handed for case 2. So in case 2 imagine I have a message, which has 1 full block of n bits and the second block it does not have full n bits. So again I have to do a padding here but here I do not have to add a new complete dummy block because it suffice for me to add 1 followed by the required number of 0 to ensure that I have the second block also have n bits.

And then I use another sub key in this case, which I call as k sub 2, do the prefix free-encoding of the padded message. And then, internally, I invoke my block wise fully secure PRF which will now only have up to l number of blocks. Because I would not be adding any dummy block in case 2 here right. And whatever comes as an output that is taken as the output of my PRF F star for the input m on the key k, so these are the 2 cases here.

(Refer Slide Time: 28:08)



Now let us see how exactly that sub keys k sub 0. K sub 1, k sub 2 are derived. So remember that the overall key for my PRF F star is just 1 key k of size n bits. So the keys k 0, k 1 k 2 are derived from the actual key k for the PRF F star by a sub key generation algorithm, which will be publicly known and depending upon whether sender is in case 1 or is in case 2. It will derive the keys k 0, k 1, k 2, but depending upon whether it is case 1 or case 2, it is either going to use k 0, k 1 or it is going to use k 0, k 2 right.

Now how the receiver is going to perform the operation. So imagine a message and a corresponding tag is sent to the receiving end and a receiver has to verify, whether the message is the right, whether the tag t is the right tag on the message m or not. And imagine that the receiver also have the same key k. So what the receiver is going to first do is this receiver is going to derive the sub keys k sub 0, k sub 1 and k sub 2.

And then it has to remove the padding right because remember, receiver is to first find out what exactly was the padding that sender has performed at it is end. So to remove the padding, what is receiver is going to do is it can start parsing the message from the right position to the left position and it keep and it can make a complete scan. And it can stop scanning as soon as it encounters the first one when it is scanning from right to left as soon as it encounters the first one.

That it knows that is the padding which has been done and it can remove and throw off the padding. And that can tell the receiver whether the receiver should operate as per case 1 or as per case 2. And I claim that this is an unambiguous padding namely the receiver strategy of scanning the message from the right position to the left position and looking for the first occurrence of the 1 and stripping of the first occurrence of one followed by all subsequent 0s is indeed and unambiguous padding for the receiver.

Because if indeed we were in case 1 that means if sender has actually added a full dummy block. Then indeed the first occurrence of 1 when receiver will do the scanning will be when it is done with the entire last block, whereas if the sender would have been in case 2, for the receiver the first occurrence of the 1 will be in the last block itself and that a toss that it is in case 2.

So once a receivers identifies whether it is in case 1 or whether it is case 2, depending upon the required case it can verify that tag part, verify the t part or the tag for the message that it has received. Either by operating the PRF F star with the sub keys K sub 0, k sub 1 or with the sub keys k sub 0 and k sub 2 right. So that brings me to the end of this lecture what we have done in this lecture is we have seen how to construct a message authentication codes for arbitrary long messages using fixed size PRF.

And the approach for the construction is that we try to design secure PRF which can take arbitrary sequence arbitrary bit string as input and give you a fixed size output. If you can construct such PRFs which can operate over arbitrary sequence of bits and gives you a fixed size output. Then using such PRF we can easily get a message authentication code which can operate on which bit strings as the input.

And we have seen a candidate construction for a PRF operating on a sequence of bits, namely the CBC PRF. And we have seen several ways of constructing that CBC PRF namely, we have seen first how to construct a prefix free block wise PRF prefix free secure block wise PRF, which is secure against a weaker adversary. And then by applying different mechanisms namely encryption, deterministic, prefix free-encoding, randomized prefix free-encoding.

We convert that weaker form of PRF, namely which is secure only against a prefix free secure adversary into a fully secure PRF, which is secure even against an adversary, which can make queries which does not constitute a prefix preset. And then finally, we construct or convert this block wise fully secure PRF into bit wise secure PRF using the CBC mode right. So now we have a tool for authenticating the messages and even to verify whether the received message has been received correctly or not.

Namely if we have a sender and a receiver who have a shared key k and if we have a message authentication code which can give you a fixed set size tag for arbitrary long messages. Then to authenticate the message sender can just compute a short or fixed length tag for that message. And along with the message the tag can be communicated to the receiver, when receiver receives that tag and if the tag gets successfully verified with a key k.

Then that gives the guarantee to the receiver that it has originated from the same person who has the same key k with which the tag verification is successful. So that is how the problem of authenticity and integrity are solved right. I hope you enjoyed this lecture thank you.