

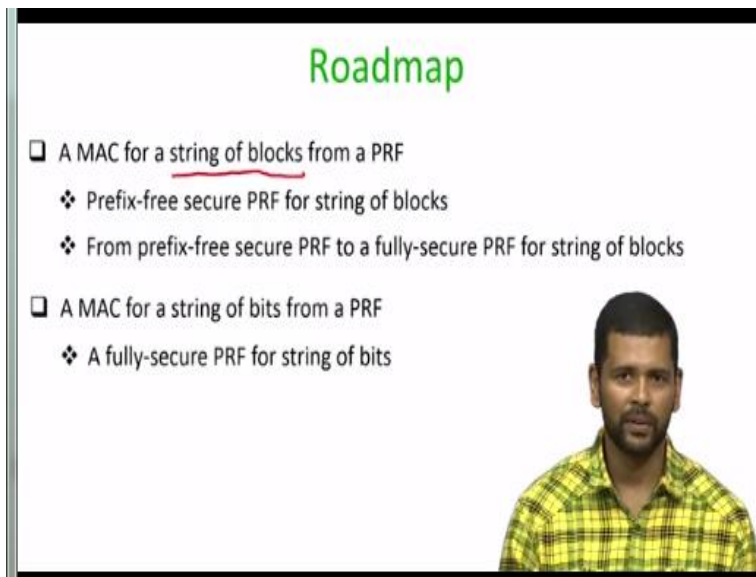
Foundations of Cryptography
Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
International Institute of Information Technology-Bengaluru

Lecture-23
Message Authentication for Long Messages Part I

Hello everyone, welcome to lecture 22, just to recap , in the last lecture we started discussing about the problems of message authentication and message integrity, where the goal of the receiver is to verify whether the message that it is receiving is coming from the right sender or not. And to detect whether the message that is received by the receiver is indeed the correct message or not.

And we introduced a cryptographic primitive called message authentication code and we have seen the security definitions of messages authentication code. And we also discussed how to construct message authentication code for fixed length messages using pseudo random functions. So in this lecture, we will continue the discussion on message authentication codes.

(Refer Slide Time: 01:12)



The slide is titled "Roadmap" in green text. It contains a list of topics to be covered, organized into two main sections. The first section is "A MAC for a string of blocks from a PRF", which includes two sub-points: "Prefix-free secure PRF for string of blocks" and "From prefix-free secure PRF to a fully-secure PRF for string of blocks". The second section is "A MAC for a string of bits from a PRF", which includes one sub-point: "A fully-secure PRF for string of bits". A small video inset of Prof. Dr. Ashish Choudhury is visible in the bottom right corner of the slide.

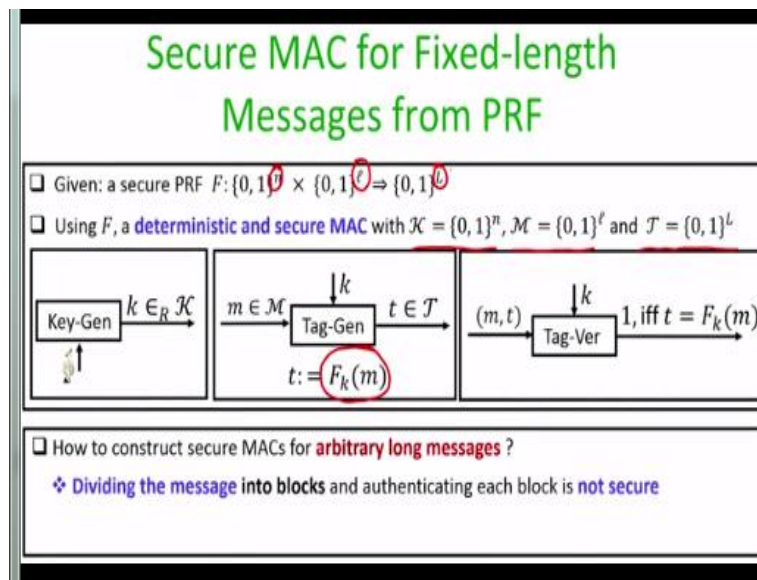
- ❑ A MAC for a string of blocks from a PRF
 - ❖ Prefix-free secure PRF for string of blocks
 - ❖ From prefix-free secure PRF to a fully-secure PRF for string of blocks
- ❑ A MAC for a string of bits from a PRF
 - ❖ A fully-secure PRF for string of bits

And what we will do is we will discuss how to construct message authentication codes for arbitrary bit strings. So we will begin with discussing how to construct a message authentication code for string of blocks and this will be further done in 2 stages. We will first

see how to construct prefix free secure pseudo random functions for string of blocks. And then we will see how to convert any prefix free secure pseudo random function for string of blocks to fully secure PRF for string of blocks.

And once we have fully secure PRF for string of blocks, we will see how to construct pseudo random functions for arbitrary length input namely arbitrary but strings which will eventually give us message authentic code for arbitrary bit strings. So that is the plan for this lecture.

(Refer Slide Time: 02:04)

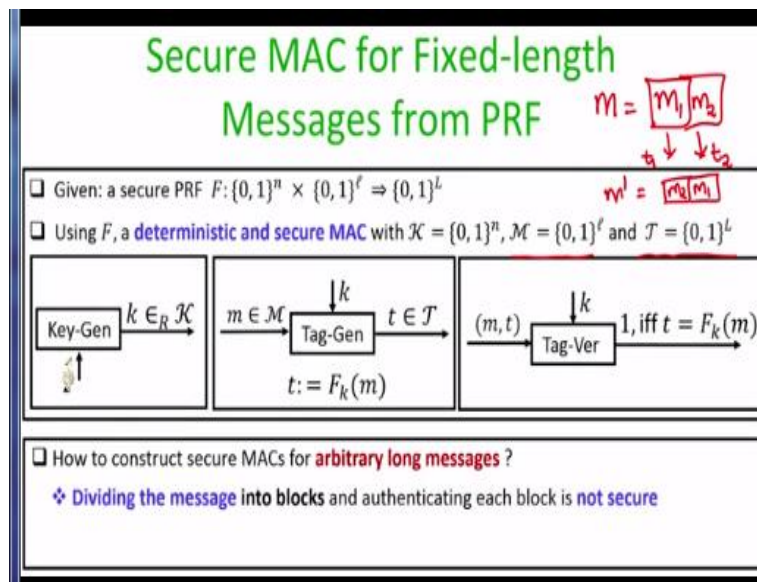


So just to recall, in the last lecture, we have seen how to construct a message authentication code for fixed length messages using pseudo random functions. So assume you are given a secure pseudo random function with n bit key l bit block length and L bit output size. Then using the pseudo random function we have seen how to construct a secure MAC where the key space is the set of n bit strings message space set of all l bit strings and tag space is a set of L bit strings.

And construction is as follows the key generation algorithm outputs key for the pseudo random function which is used to authenticate the message by computing the value of the pseudo random function under the key k with the message being the input block. And that is the tag and to verify the tag for the message m . The tag is recomputed by recomputing the value of the keyed function F_k on the input m .

And matching whether the recomputed tag matches the received tag or not. If the match happens then the output is 1, otherwise the output is 0. Now our goal is to construct secure MACs for arbitrary long messages, because in practice there is no restriction that the message size should be of L bit, it could be less than L bit, it could be larger than L bits and so on.

(Refer Slide Time: 03:29)



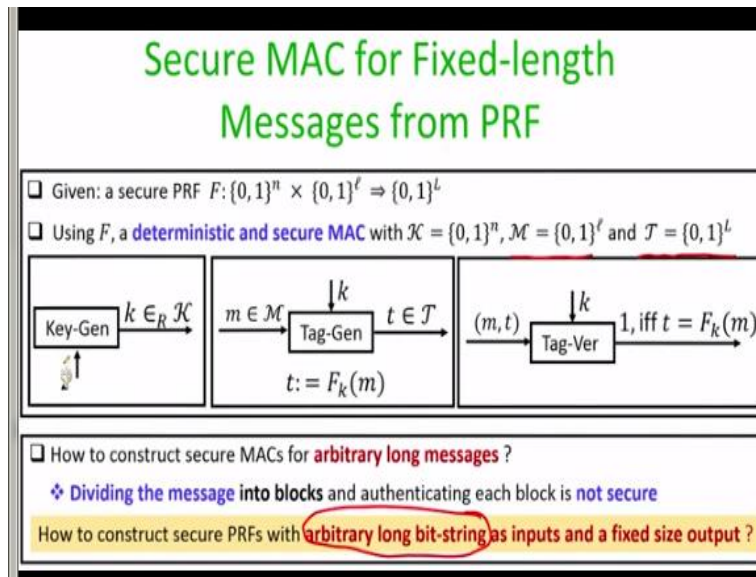
And it turns out that unlike CPA secure encryption schemes, where if we have a CPA secure encryption scheme for encrypting L bit messages. And if we want to achieve CPA security for encrypting arbitrary long messages, then we had seen that it suffices to divide the larger message into several blocks of L bits, and encrypt each block of L bits using an instance of the fixed length encryption process.

But that process of dividing the message and authenticating each block using this fixed sized MAC is not going to give you a secure MAC construction. Specifically assume that you have a message m right consisting of 2 blocks of L bits. And you divide the message m into blocks m_1 and m_2 and say you authenticate m_1 using this fixed length MAC procedure and you authenticate independently m_2 using the same key k using this fixed length MAC procedure.

Then an adversary who has seen the tag t_1, t_2 for a message m can easily produce a MAC for a new message m' consisting of block m_2 followed by m_1 right. And that will be a forgery from

the viewpoint of the adversary. So it is not straightforward to just take this MAC construction for fixed length messages and use it to construct a MAC for authenticating arbitrary long messages.

(Refer Slide Time: 04:49)

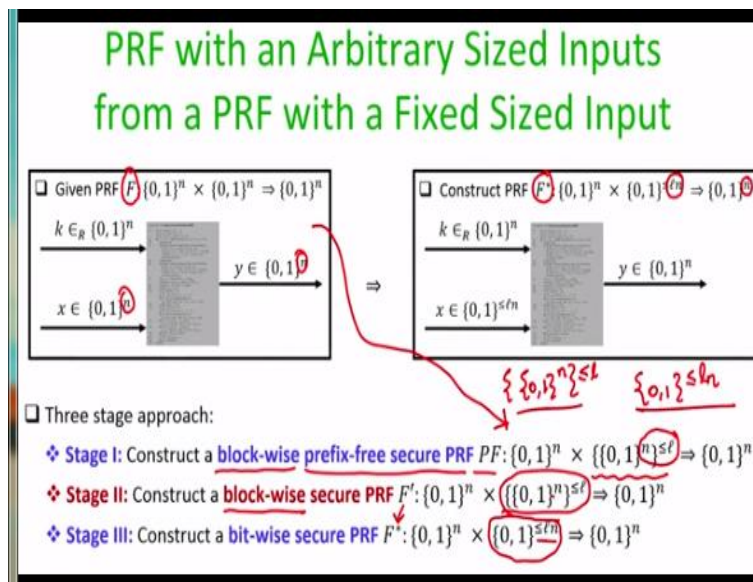


The construction is really going to be a challenging task. So the idea behind constructing message authentication codes for arbitrary long messages is to look for, it should design secure pseudo random functions with arbitrary long bit strings, but fixed size output. Because if we can construct pseudo random functions, which can support any arbitrary long bit string as an input and gives a fixed size output, then using such a pseudo random function, we can easily construct a MAC for arbitrary long messages, what you basically have to do is you just have to invoke that pseudo random function on the message, which could be of any length to produce the tag. And as per the idea that we had used for constructing the pseudo MAC for a fixed length message, the corresponding construction will be a secure construction.

So the whole problem of designing a secure MAC for arbitrary long messages boils down to the problem of how to construct pseudo random functions for arbitrary long bit strings. And that is what will be the focus of this discussion. So the reference material for today's discussion will be the chapter from the book draft by Boneh and Shoup and most of the proofs for the constructions that we are going to discuss in today's lecture will be skipped because the proofs are really advanced.

We will be just giving, we will be just discussing the overview of the high level ideas behind the security proofs of the constructions that we are going to discuss in this lecture. But if you are interested to see that exact proofs then you can refer to the lecture given in the book draft by Boneh and Shoup right.

(Refer Slide Time: 06:36)



So let us start with the construction of a PRF for arbitrary size inputs from a PRF with fixed size input. So what you are given is assume you are given as PRF, whose block length is n bits and output is also of n bits. And using this, your goal is to basically construct another keyed PRF which I denote as F^* . And which can support input of any length up to $(l * n)$ bits. And it will give you a fixed output of size n bits.

So as you can see that the output of the keyed PRF F^* which we are interested to construct is independent of the value l right. So it does not matter whether your l is 1, l is 2 and so on the output size is always fixed, namely n bits. So our goal is how to construct this F^* given this keyed pseudo random function F . And we will design F^* from F using a 3 stage approach.

In stage 1 what we will do is, we will first construct a PRF which would not be operating on arbitrary length input, but it will be operating on arbitrary sequence of blocks. Namely it can compute the output for any number of up to l number of blocks, where each block will consist

of n bit of input. And it always gives you an output of n bit and we call this PRF as block wise prefix free secure PRF.

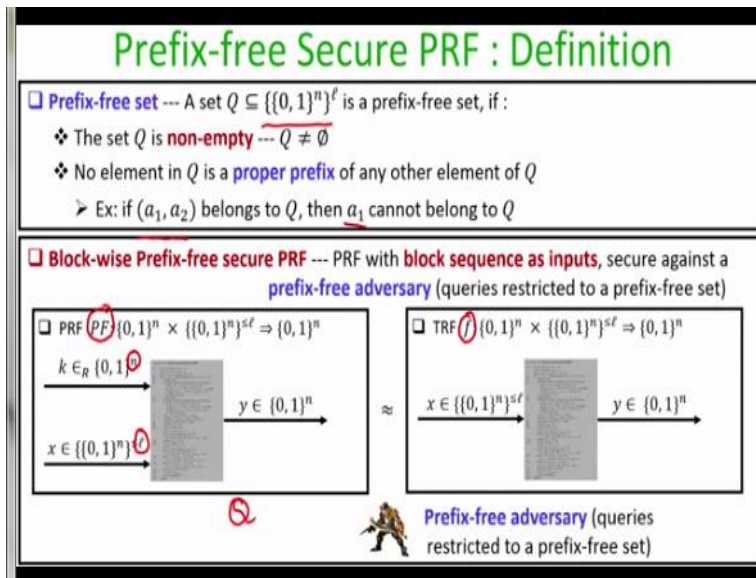
It will be clear very soon what exactly we mean by the term prefix free secure PRF. But important thing is that it does not support inputs of arbitrary length, but rather it supports inputs which are actually parsed as sequence of several blocks. And it can support inputs consisting of up to l blocks, where each block further consists of n bits of string. So that is denoted by this notation $\{\{0,1\}^n\}^{\leq l}$.

That means each block is of size n bits and there can be up to l such blocks. So the stage 1 will be how to go from this fixed size PRF F to this prefix free secure PRF PF . Then once we have the construction PF , which is block wise prefix free secure PRF. In stage 2, we take this prefix free secure PF and we construct a fully secure PRF, but which is still block wise. That means it can take inputs consisting of upto l blocks, where each block is of size n bits and the output is of size n bits.

And then finally in stage 3, once we have the construction F' , we construct the actual PRF F^* which we are interested to construct, which can take any input any arbitrary bit string as input of length of up to $l \cdot n$ bits. So remember there, so notice that there is a difference between the notation where this notation and the notation where binary string up to length $l \cdot n$ bits, so when I say input of the form $\{0,1\}^{\leq l \cdot n}$ that means the input is an arbitrary bits string whose length is $l \cdot n$. But when I say that I have an input of this form $\{\{0,1\}^n\}^{\leq l}$, that means my input consist of several blocks, namely, up to l blocks, where each block is exactly of size n bits. So there is a difference between this notation and this notation. So that is the 3 stage approach that we are going to follow to construct our required PRF F^* from the fixed length PRF F right.

And each of the stages is really interesting and subtle and we will be skipping the proofs as I said earlier, if you want to see the proof, you can see the chapter in the book draft by Boneh and Shoup.

(Refer Slide Time: 10:37)



So let us begin with the construction of prefix free secure PRF and for that, let me first define what exactly we mean by prefix-free set. So a set Q which is a subset of set of all blocks, up to which is a subset of sequence of blocks up to size l blocks, where each block is consisting of n bits is called a prefix-free set. If the following conditions hold, first of all the set Q should be non empty.

And the second property is that no element in the set Q should be a proper prefix of any other element of the set Q , what this means is that if we have an input member in the set Q say the sequence (a_1, a_2) where a_1 is 1 block and a_2 is another block each of size n bits. Then we cannot have the input a_1 present in the set Q , because the block input a_1 is a proper subset of the sequence of block inputs a_1, a_2 . However, we can have the input a_2 present in the block Q because a_2 is not a prefix of the block sequence (a_1, a_2) .

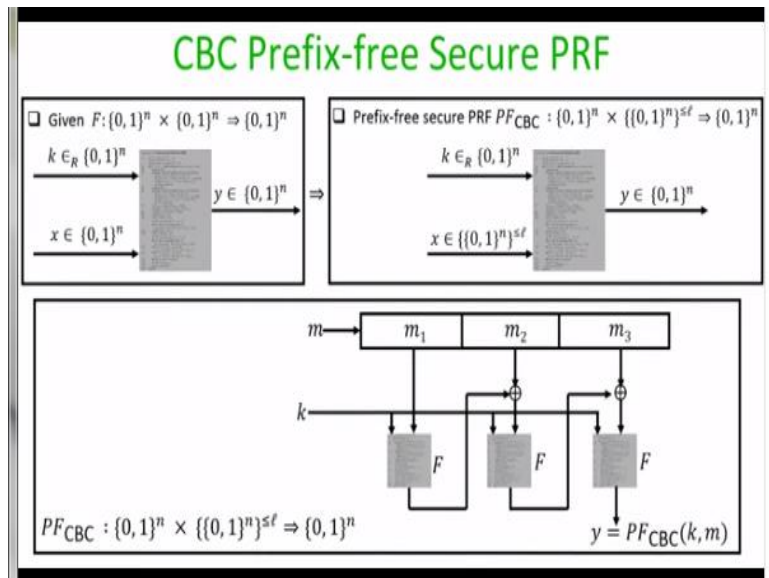
So that is the definition of a prefix free set, right. So now what exactly we mean by a block wise prefix free-secure PRF. So on a very high level it is a PRF right which take inputs as a block sequence and there could be up to l number of such blocks. And that PRF will be secure against a weaker adversary which we call us prefix free adversary. Namely, it will be secure against an adversary or a distinguisher whose queries are restricted to a prefix free-set.

So what exactly we mean by that, so syntactically block wise prefix free PRF will be a keyed function and it will take a key of size say n bits. And it takes a sequence of blocks as inputs, so it can take up to l blocks here, where each block is a fixed size namely n bits. And it gives you a fixed size output say n bits and a security property that we want from this block wise prefix-free secure PRF is that, if you consider a truly random function which also takes input a sequence of blocks say up to l blocks, where each block is of size exactly n bits. And gives you an output of size n bits such that the function f is a truly random function right, it is a unkeyed function. Then what we mean by a prefix free secure PRF is that the behavior of this keyed PRF should be indistinguishable from the behavior of this truly random function f against a distinguisher, who is restricted to make queries only from a prefix free-set.

That means this distinguisher cannot ask for the value of the function on block sequence input (a_1, a_2) as well as a_1 that means whatever queries this distinguisher is going to ask. In the PRF indistinguishability game, if we denote it by Q , then the set Q of queries for the distinguisher should constitute a prefix-free set. So that is why this distinguisher is a weaker distinguisher or a weaker adversary.

Because in the actual PRF security definition, there is absolutely no restriction put on the nature of queries which the distinguisher can ask during the game. It could be a prefix-free set, it may not be a prefix-free set. But when I say we are designing a prefix-free secure PRF then it will be secure only against an adversary, whose queries constitute a prefix free set right.

(Refer Slide Time: 14:23)



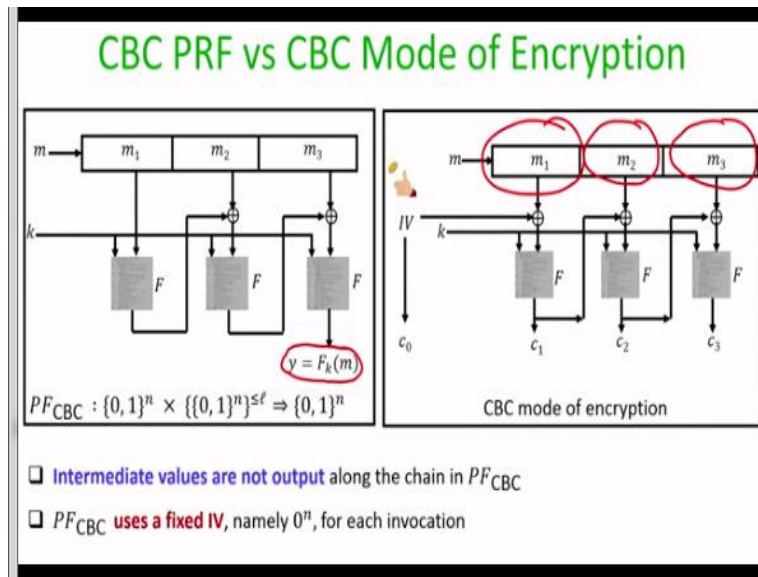
So that is the definition of block wise PRF which is prefix-free secure and now let us see a candidate construction which we call a CBC prefix free secure PRF. And the construction is as follows. So we are given a keyed PRF of fixed size input and fixed size output. And using that our goal is to construct a PRF which is a prefix-free secure PRF right which can take a sequence of blocks as input say up to l blocks each of size n bits and gives you a fixed size output.

And a way we do it is using this CBC mode of encryption that we had seen earlier with some differences. So let me demonstrate how exactly this CBC prefix-free secure PRF operates. For demonstration purpose, assume that you have a message m consisting of up to 3 blocks right. Now to compute the output of the CBC prefix-free secure PRF for this input, what we do is the following.

Since we have 3 blocks we need to invoke the fixed size PRF 3 times with the same key k . And the first invocation of the PRF will be with m_1 as the input and whatever is the output that serves as the input for the next invocation, where it is XORed with the next block input, namely m_2 and a XOR of the m_2 and the previous output of the PRF is fed as the block input for the second invocation, and then we continue the chaining process.

And finally, the outcome of the last invocation of the fixed length PRF F is treated as the output of the prefix-free CBC secure PRF for the message m . That is a way, this CBC prefix-free secure PRF operates ok.

(Refer Slide Time: 16:14)



So there are certain differences between the CBC PRF that we have designed just now and CBC mode of encryption that we had discussed in one of our earlier lectures. So on your left hand side, you have the CBC PRF mode and on your right hand side you have the CBC mode of encryption right. For comparison purpose, what I am considering is I am considering a message block consisting of 3 blocks and how the output of the prefix- free CBC PRF will be computed on that message.

And on the right hand side part I have the same message and I am demonstrating how exactly the CBC mode of encryption will be operated on the same message right. So the differences between these 2 primitives are as follows. First of all, when we see the CBC PRF, then the intermediate values they are not output at all right, it is only the outcome of the final invocation of F , which is output as the overall output of the CBC PRF.

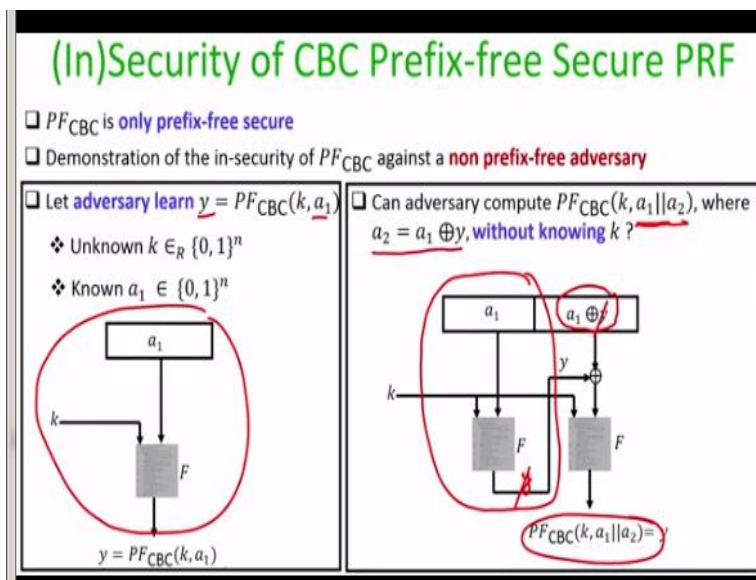
Whereas in the CBC mode of encryption, each of the intermediate outcomes of the F are also part of the overall output. This is because in the CBC mode of encryption, the goal is to encrypt each of the individual blocks. So that is why we need to definitely output the intermediate

outputs of the PRF. But when we consider the CBC PRF, our goal is to compute a fixed size output for the overall message. So that is why it is not necessary to output the intermediate invocations of the PRF, so that is the first difference.

The second difference is that if we consider the CBC PRF then it uses a fixed IV, namely in the picture there is no IV at all. But you can always imagine that the IV is set to all 0s which is publicly known. But when it comes to a CBC mode of encryption, then there is an IV and which is randomly selected for each instance of the CBC mode of encryption. So you can imagine that CBC PRF uses a deterministic IV, which is always fixed for every message.

But when it comes to the CBC mode of encryption, IV is selected independently for each invocation.

(Refer Slide Time: 18:28)



So these are the 2 differences and it turns out that if you actually modified a CBC PRF by also outputting the intermediate outputs of the PRF, then the overall PRF is not a secure PRF. Also instead of using a fixed IV, if you start using a random IV, there is no guarantee that overall the CBC PRF is a indeed a prefix free secure PRF right. So that is how the CBC PRF operates and I would not be going into the security proof that why exactly this CBC PRF is indeed a prefix-free PRF.

But let me give you an intuition that why it may become insecure if we do not restrict our adversary from making queries which constitute a prefix-free set right. So what I am going to do is I am going to demonstrate the insecurity of the CBC prefix-free secure PRF against an adversary which can make queries which does not constitute a prefix-free set. So imagine that adversary, there is an adversary which queries for a block input a_1 and gets the output y under an unknown key k which is not known to the attacker right.

So the way output y would have been computed is as follows. Since we have only one block, basically we would have evaluated the fixed length PRF F on the input a_1 with the key k and that is how the output y would be computed. Now imagine adversary has a new block new input consisting of 2 blocks, say block a_1 followed by a block a_2 , where a_2 is a special block here, a_2 is related to the block a_1 and the output y which is the value of the prefix-free PRF on the block input a_1 .

And now let us ask what exactly will be the value of this prefix-free CBC PRF on the input (a_1, a_2) right. So this is how the output of the CBC PRF on the input (a_1, a_2) will be computed right. We will have 2 invocations of the fixed length PRF, we will do the chaining and the final output will be this: $PF_{CBC}(k, a_1 || a_2)$. Now what exactly is the outcome of the first invocation of the PRF F here it is y , because that is what is the structure right.

So you can see that if I just focus on this part here, this part is nothing but the computation of the CBC PRF on a different input consisting of just a block a_1 . And we know that is nothing but y and now this y is fed as the input for the second invocation, where this y is XORed with the next block here. And the next block is $a_1 \text{ XOR } y$, so the effect of this y and y cancels out.

And what basically we are doing here is we are actually invoking the fixed length PRF on the input a_1 . And adversary already knows that the value of this PRF under the unknown key k but with the known input a_1 is going to give you the output y .

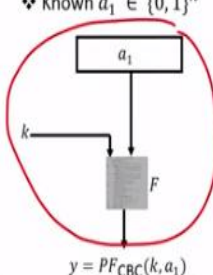
(Refer Slide Time: 21:48)

(In)Security of CBC Prefix-free Secure PRF

- ❑ PF_{CBC} is only prefix-free secure
- ❑ Demonstration of the in-security of PF_{CBC} against a non prefix-free adversary

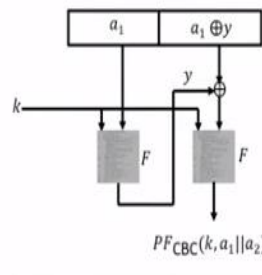
❑ Let adversary learn $y = PF_{CBC}(k, a_1)$

- ❖ Unknown $k \in_R \{0, 1\}^n$
- ❖ Known $a_1 \in \{0, 1\}^n$



$y = PF_{CBC}(k, a_1)$

❑ Can adversary compute $PF_{CBC}(k, a_1 || a_2)$, where $a_2 = a_1 \oplus y$, without knowing k ?



$PF_{CBC}(k, a_1 || a_2) = y$

So adversary already knows that the output of the CBC PRF on the input sequence (a_1, a_2) will give you y if thus block a_2 satisfies this relationship. And that gives him an advantage or a strategy to distinguish this CBC PRF from a truly random function right. So what the adversary basically can do is, it can first ask for the function output on the block input a_1 . And then it can ask for the function output on the block sequence of block inputs (a_1, a_2) .

And it can compare it with y , if it sees the output of the function input on the sequence (a_1, a_2) to be y . Then it knows that it is interacting with a prefix-free CBC PRF and if the output is not y that it means that it is interacting with a truly random function. And it can be formally proved that the distinguishing advantage of the distinguisher is significant right. So that means if we are allowing adversary to query, if the set of queries which the adversary can ask does not constitute a prefix-free set.

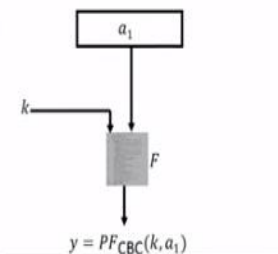
(Refer Slide Time: 22:48)

(In)Security of CBC Prefix-free Secure PRF

- ❑ PF_{CBC} is **only** prefix-free secure
- ❑ Demonstration of the in-security of PF_{CBC} against a **non prefix-free adversary**

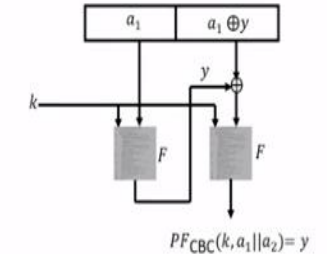
❑ Let **adversary learn** $y = PF_{CBC}(k, a_1)$

- ❖ Unknown $k \in_R \{0, 1\}^n$
- ❖ Known $a_1 \in \{0, 1\}^n$



$y = PF_{CBC}(k, a_1)$

❑ Can adversary compute $PF_{CBC}(k, a_1 || a_2)$, where $a_2 = a_1 \oplus y$, **without knowing k** ?



$PF_{CBC}(k, a_1 || a_2) = y$

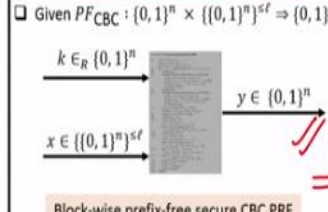
Then definitely the construction that we have seen now is not secure right. Because in this particular example he is asked to query for a_1 and then he is asked to query for also the input (a_1, a_2) . And clearly a_1 is a prefix of the set (a_1, a_2) but the definition of prefix-free secure PRF is that it is secure only against an adversary who cannot query for both a_1 as well as the input sequence (a_1, a_2) right.

So that is the basic intuition that why this chaining PRF is secure against a prefix free adversary. If we do not put that restriction then definitely it is distinguishable from a truly random function.

(Refer Slide Time: 23:34)

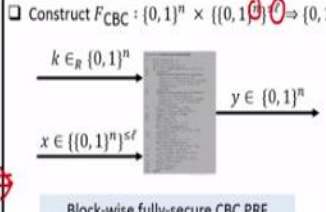
From Prefix-free Secure CBC to Fully-secure CBC

❑ Given $PF_{CBC} : \{0, 1\}^n \times \{ \{0, 1\}^n \}^{s\ell} \Rightarrow \{0, 1\}^n$



Block-wise prefix-free secure CBC PRF

❑ Construct $F_{CBC} : \{0, 1\}^n \times \{ \{0, 1\}^n \}^{s\ell} \Rightarrow \{0, 1\}^n$



Block-wise fully-secure CBC PRF

❑ **Three different approaches** with different tradeoffs

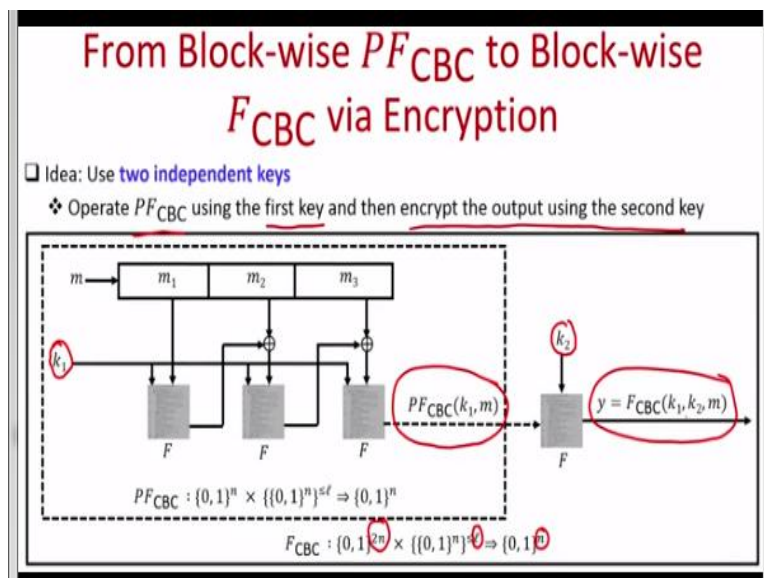
- ❖ Encrypted CBC ✓✓
- ❖ Deterministic prefix-free encoding }
- ❖ Randomized prefix-free encoding }

So now we have a candidate block wise prefix-free secure PRF. And what we do is now we will go to the stage 2, where we will convert this block wise prefix-free secure PRF to get a fully secure PRF, where there is no restriction whatsoever on the queries which the distinguisher can make in the PRF indistinguishability game right. So you are given this we have already seen a candidate for this namely the CBC PRF block wise.

And our goal is to construct a fully secure block wise PRF, which can take sequence of blocks as inputs, say up to l blocks where each block is of fixed size, say n bits. And the output is fixed size output of size n bits. And there are several approaches for doing this, there are several approaches from going to this prefix-free secure PRF to a fully secure PRF and each approach has a different trade off.

So the first approach is an encrypted CBC PRF and the remaining 2 approaches are deterministic prefix-free encoding and randomized prefix encoding. So the last two method what it does is basically it encodes the input of your block wise prefix-free secure PRF, in such a way that the resultant encoded inputs constitute a prefix-free set, whereas the encrypted CBC uses 2 keys., where using the first key we operate the prefix-free PRF. And then the output is again encrypted using the second key which ensures that adversary is confused.

(Refer Slide Time: 25:12)



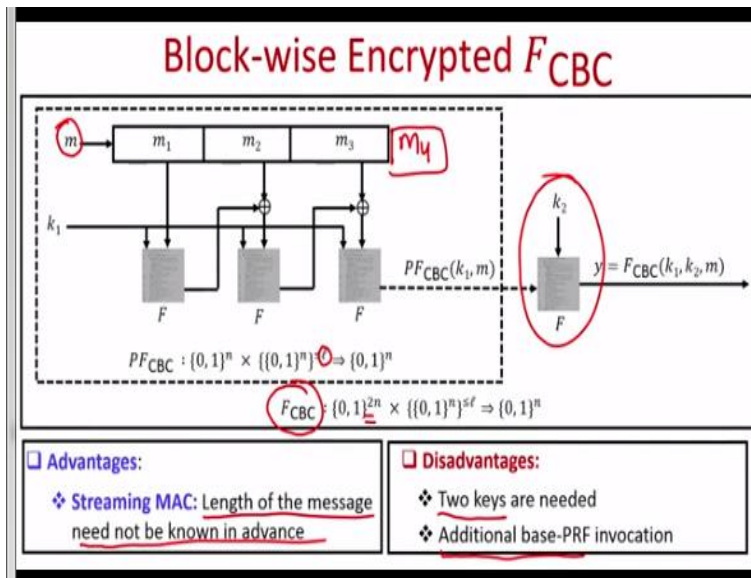
So let us go deeper into the different approaches, so let us first see how we construct the block wise prefix-free secure CBC to a fully secure CBC using encryption. And the idea here is to use 2 independent keys, where using the first key we operate the prefix-free block wise CBC and using the second key we encrypt the output of the block wise CBC. More specifically what we are going to do is, we are going to construct a new PRF which I denote as F_{CBC} which is fully secure.

There is no restriction on the distinguisher it can make queries which maybe prefix-free set or which may not constitute a prefix-free set and so on. So the construction is going to take 2 keys and the PRF can support a sequence of blocks, say up to l blocks. Each block consisting of fixed n bits and I stress that here throughout this discussion this l is some polynomial function of your security parameter.

And the output of this block wise PRF is going to be of n bits right. So how exactly the operation will happen here, since the key is of size $2n$ bits, we parse it as sequence of 2 blocks of n bits each and we call the first part of the key to be k_1 . And with k_1 , we actually first run the prefix-free CBC right which operates block wise and this will be the output of the prefix-free CBC PRF with respect to the key k_1 .

And once we obtain the output, we use the remaining part of the key which is also of n bits and independent of k_1 . And we use this key to again encrypt the output that we have obtained from the prefix-free CBC PRF and that will be the overall outcome of this fully secure CBC PRF with respect to the key k_1, k_2 right.

(Refer Slide Time: 27:14)



So there are several advantages as well as disadvantages for this block wise encrypted CBC PRF. The advantage here is that this PRF constitutes what we call as a streaming PRF and which further gives us what we call as a streaming MAC. So remember, our goal is to basically construct PRF supporting arbitrary block size input; our goal is to construct PRF, which can take inputs of arbitrary size.

Right now what we are aiming is to construct PRF which can take sequence of blocks as input and gives you a fixed size output. And once we have such PRF, we can always use those PRF to obtain the corresponding secure MACs. Namely a MAC which can take inputs; which can take sequence of blocks as input and give you a fixed size tag. So the advantage of this encrypted CBC PRF is that it constitutes what we call as a streaming MAC.

Namely in this construction the length of the message need not be known in advance right. That means imagine a scenario where a sender is continuously sending message packets to the receiver. And it does not know exactly what is going to be the length of the message m that means it does not know well in advance up to how many blocks of l bits will be there in the message m , what it knows is that maximum number of blocks that can be there in the message m is l , where l is publicly known.

But apart from that it has no knowledge whatsoever about the exact number of blocks. So if you see the way this encrypted CBC PRF is operating, the length of the message is not at all useful here. So as and when new blocks are coming right; what the sender can do is basically it can compute the corresponding output of the prefix-free CBC PRF.

And once the message is over, that means once the last block of the message is coming to the sender, it can compute a final invocation of the PRF with respect to the key k_2 . So that is why it is constitutes what we call as a streaming MAC because the length of the message need not be known in advance here. The disadvantage in this construction is that we now need to operate with 2 keys.

That means the overall key for the CBC block wise PRF actually consists of of 2 independent chunks of n bits. And we also need an additional PRF invocation namely the final PRF invocation apart from the number of PRF invocations which we are invoking internally as part of the prefix-free CBC PRF. So these are the 2 disadvantages, and we also have a advantage here right.

So you have the trade off here. If you want a streaming MAC, then this really is a very good construction. But you need to pay something for that namely you need to operate with 2 keys and you should be willing to have an additional PRF invocation here right. I hope you enjoyed this lecture, thank you.