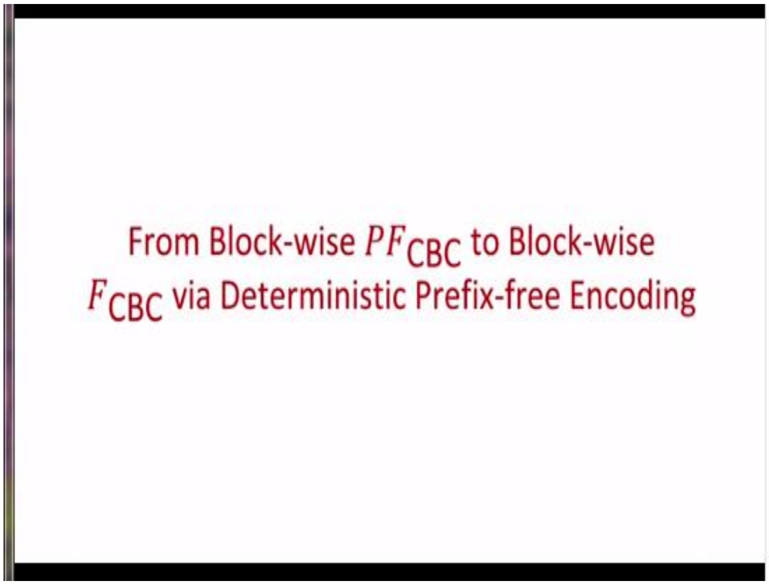
Foundations of Cryptography
Prof. Dr. Ashish Choudhury
(Former) Infosys Foundation Career Development Chair Professor
Indian Institute of Technology-Bengaluru

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 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.

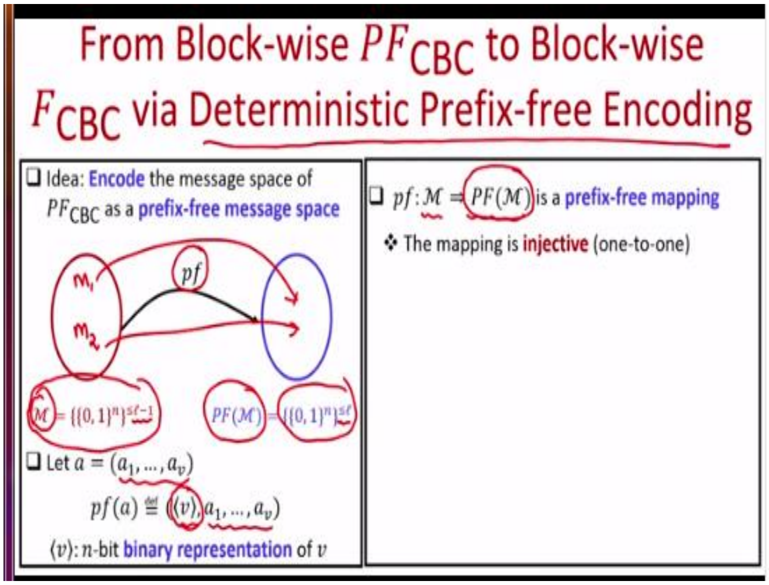
(Refer Slide Time: 00:49)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Deterministic Prefix-free Encoding

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

(Refer Slide Time: 00:59)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Deterministic Prefix-free Encoding

❑ Idea: Encode the message space of PF_{CBC} as a prefix-free message space

$\mathcal{M} = \{\{0,1\}^n\}^{\leq \ell - 1}$ $\quad PF(\mathcal{M}) = \{\{0,1\}^n\}^{\leq \ell}$

❑ Let $a = (a_1, \dots, a_v)$

$pf(a) \stackrel{\text{def}}{=} (\langle v \rangle, a_1, \dots, a_v)$

$\langle v \rangle$: n-bit binary representation of v

❑ $pf : \mathcal{M} \rightarrow PF(\mathcal{M})$ is a prefix-free mapping

❖ The mapping is injective (one-to-one)

And this will be through some encoding. And the basic idea here is that, we apply some publicly known encoding function to the message space of the block-wise prefix-free CBC PRF, so that the encoded message space constitutes a prefix-free message space. And what basically it means is the following: imagine you have a message space for the prefix-free PRF, consisting of up to $\ell - 1$ blocks, each of size n bits. What we will do is, we will apply some mapping or encoding function which I denote as pf and the encoded message space will now consist of a sequence of block up to ℓ blocks, each of size n bits. And the way mapping is done here 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.

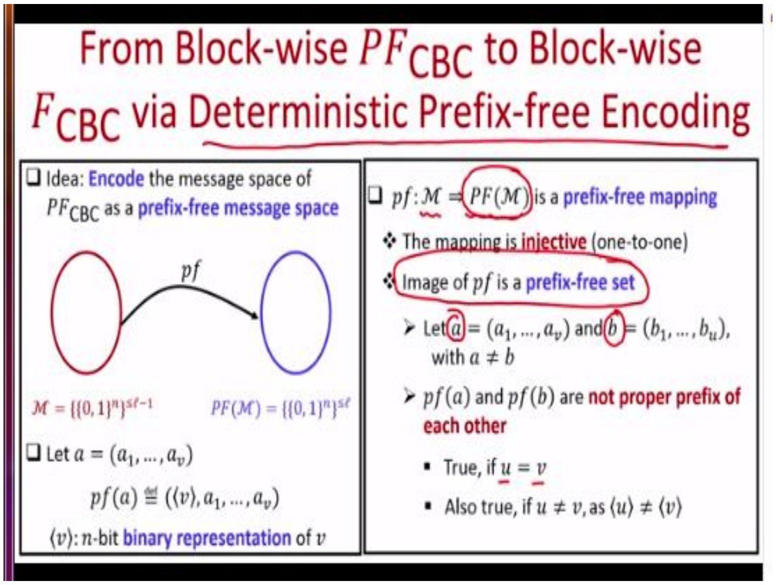
So, the way this deterministic encoding mechanism works is as follows. So say you have a sequence of input blocks a, the corresponding encoded input a will be as follows: all the message blocks of a are copied as it is in the encoded output and apart from that we now introduce an additional block here, 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 of v. So that is why you can see that the input could consist of up to $\ell - 1$ blocks. And since we are now introducing an additional block in the encoded output, that is why the output now can consist of up to 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 original message space, we are now going to operate it over the encoded message space. And since the encoded message space will constitute a prefix-free set, the distinguisher cannot issue queries, which constitute a prefix-free set 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 constitutes a prefix free set; that is what we want to prove. So first of all it is easy to see that the mapping is injective, namely it is one-to-one. That means if you have 2 inputs, m_1 and m_2 consisting of sequence of blocks, up to $\ell - 1$ 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. 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 one-to-one mapping.

(Refer Slide Time: 04:25)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Deterministic Prefix-free Encoding

- ❑ Idea: Encode the message space of PF_{CBC} as a prefix-free message space

pf
$\mathcal{M} = \{\{0,1\}^n\}^{\leq \ell - 1}$
$PF(\mathcal{M}) = \{\{0,1\}^n\}^{\leq \ell}$

- ❑ Let $a = (a_1, \dots, a_v)$

$$pf(a) \overset{\text{def}}{=} (\langle v \rangle, a_1, \dots, a_v)$$

$\langle v \rangle$: n-bit binary representation of v

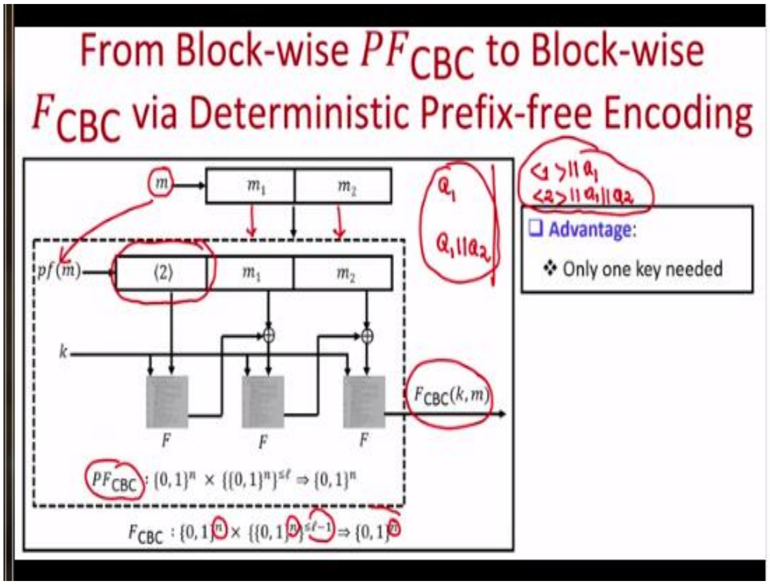
- ❑ $pf : \mathcal{M} \Rightarrow PF(\mathcal{M})$ is a prefix-free mapping
  - ❖ The mapping is injective (one-to-one)
  - ❖ Image of pf is a prefix-free set
    - ➢ Let $a = (a_1, \dots, a_v)$ and $b = (b_1, \dots, b_u)$, with $a \neq b$
    - ➢ $pf(a)$ and $pf(b)$ are not proper prefix of each other
      - ▪ True, if $u = v$
      - ▪ Also true, if $u \neq v$, as $\langle u \rangle \neq \langle v \rangle$

Now we want to prove that the image set PF, namely all the encoded messages that we get as per this mapping, constitutes a prefix-free set. 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. So my claim is that the encoded output $pf(a)$ and encoded output $pf(b)$ are not proper prefix of each other.

And if you put prove this for an arbitrary input a and an arbitrary input b, that proves that the mapping that we have considered indeed gives you a prefix-free set. 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 the same, namely if u and v are same. Because 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 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 $u = v$.

Whereas if u is not equal to v, that means the number of blocks in a and number of blocks in b are not same, 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-free set.

(Refer Slide Time: 06:21)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Deterministic Prefix-free Encoding

$PF_{\text{CBC}} : \{0,1\}^n \times \{\{0,1\}^n\}^{\leq \ell} \Rightarrow \{0,1\}^n$

$F_{\text{CBC}} : \{0,1\}^n \times \{\{0,1\}^n\}^{\leq \ell - 1} \Rightarrow \{0,1\}^n$

❑ Advantage:

❖ Only one key needed

So, now let us see how we apply this prefix-free encoding to a prefix-free secure CBC block-wise secure PRF and get a fully-secure block wise PRF. So our goal is to construct a block-wise fully secure PRF taking n-bit key and a sequence of blocks as input of up to $\ell - 1$ blocks each of size n bits, and to obtain a fixed size output of size n bits. And for demonstration, assume that we want to operate this block-wise fully secure CBC PRF on an input consisting of 2 blocks m_1 and m_2.

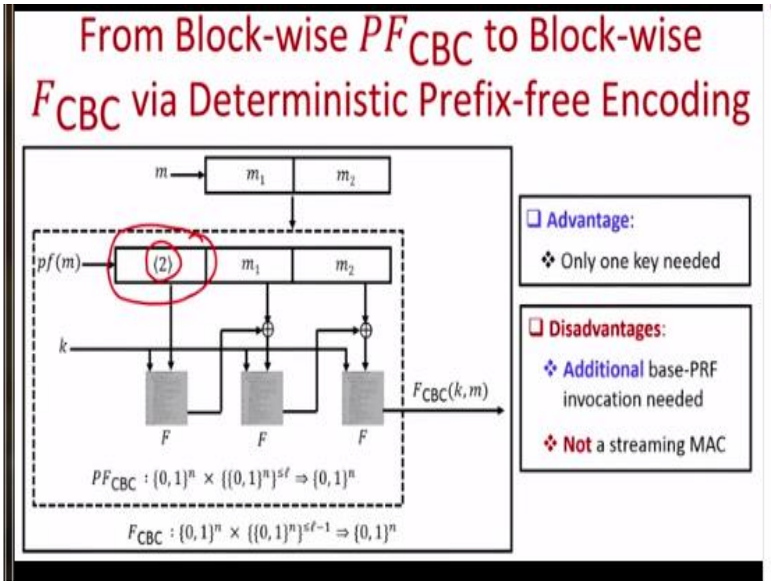
So what we will do is that we will first convert this input m by applying our deterministic prefix-free encoding. Namely this message m is converted into an encoded input. And when we convert into encoded input, then the message blocks are repeated as it is and apart from that at the beginning we now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

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.

Now why this construction is intuitively secure? Because now even if an adversary asks for the function output on an input a_1 and on an input a_1, a_2, it will be getting the output of the function on the input sequence $< 1 >, a_1$ and the output sequence $< 2 >, a_1, a_2$. 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 does not constitute a prefix-free set, basically those queries are converted into another set of queries, which is a prefix-free set and from the viewpoint of an adversary, it is now as good as he is interacting with a normal PRF.

So that is the 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. 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)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Deterministic Prefix-free Encoding

m
m_1
m_2
$pf(m)$
(2)
m_1
m_2
k
F
F
F
$F_{\mathrm{CBC}}(k, m)$
$PF_{\mathrm{CBC}} : \{0,1\}^n \times \{(\{0,1\}^n)\}^{\leq \ell} \Rightarrow \{0,1\}^n$
$F_{\mathrm{CBC}} : \{0,1\}^n \times \{(\{0,1\}^n)\}^{\leq \ell - 1} \Rightarrow \{0,1\}^n$

❑ Advantage:

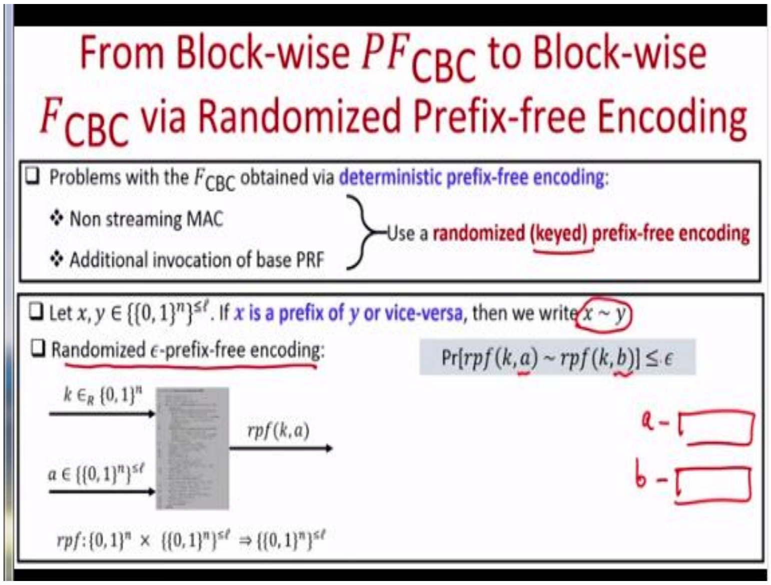
- ❖ Only one key needed

❑ Disadvantages:

- ❖ Additional base-PRF invocation needed
- ❖ Not a streaming MAC

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. We need an additional PRF invocation, 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 the 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. 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)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Randomized Prefix-free Encoding

❑ Problems with the F_{CBC} obtained via deterministic prefix-free encoding:

- ❖ Non streaming MAC
- ❖ Additional invocation of base PRF

Use a randomized (keyed) prefix-free encoding

❑ Let $x, y \in \{\{0,1\}^n\}^{\le \ell}$. If x is a prefix of y or vice-versa, then we write $x \sim y$

❑ Randomized ϵ-prefix-free encoding: $\Pr[rpf(k,a) \sim rpf(k,b)] \le \epsilon$

$k \in_R \{0,1\}^n$

$a \in \{\{0,1\}^n\}^{\le \ell}$

$rpf(k,a)$

$rpf : \{0,1\}^n \times \{\{0,1\}^n\}^{\le \ell} \Rightarrow \{\{0,1\}^n\}^{\le \ell}$

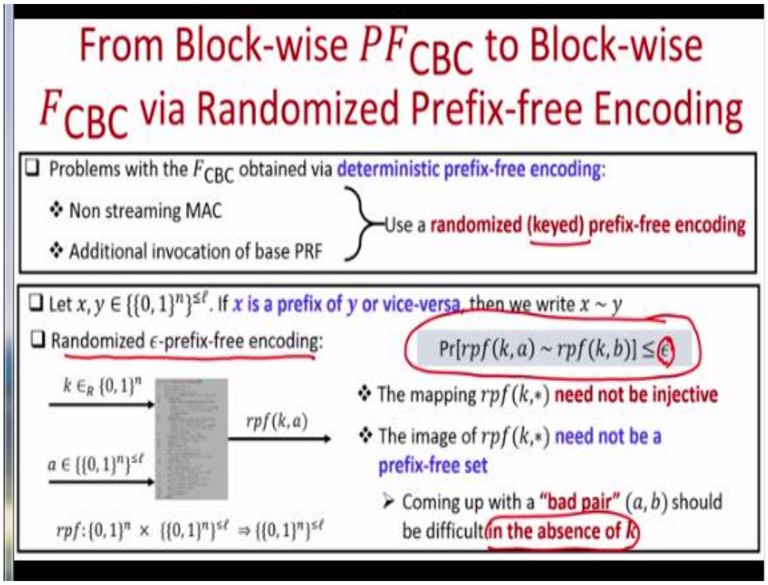
And now we see how we can go from this block-wise prefix free secure PRF to block-wise fully secured PRF using a randomized prefix free-encoding. So the reason we want to go for a randomized prefix free-encoding is that the problems that we face with the deterministic prefix free-encoding is that we 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 let us first introduce some notations here. So imagine you have 2 block sequence inputs, 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 $x{\sim}y$.

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

i. e the encoded a is a prefix of encoded b or the encoded b is a prefix of encoded a is upper bounded by ϵ. That is what I mean when I say that I have a randomized ϵ-prefix-free encoding.

(Refer Slide Time: 12:44)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Randomized Prefix-free Encoding

❑ Problems with the F_{CBC} obtained via deterministic prefix-free encoding:

❖ Non streaming MAC

❖ Additional invocation of base PRF

Use a randomized (keyed) prefix-free encoding

❑ Let $x, y \in \{\{0,1\}^n\}^{\leq \ell}$. If x is a prefix of y or vice-versa, then we write $x \sim y$

❑ Randomized ϵ-prefix-free encoding:

$\Pr[rpf(k,a) \sim rpf(k,b)] \leq \epsilon$

$rpf : \{0,1\}^n \times \{\{0,1\}^n\}^{\leq \ell} \Rightarrow \{\{0,1\}^n\}^{\leq \ell}$

❖ The mapping $rpf(k,*)$ need not be injective

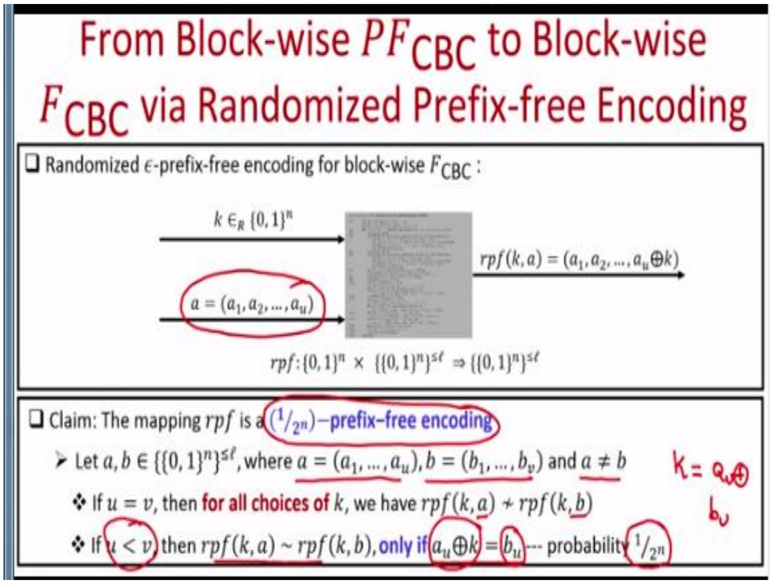
❖ The image of $rpf(k,*)$ need not be a prefix-free set

➢ Coming up with a "bad pair" (a,b) should be difficult in the absence of k

So basically the idea here is that, if there is a computationally bounded adversary, and if it does not know the value of the key k with which we are actually doing the encoding, then for the adversary it is very difficult to come up with a pair of inputs (a, b) consisting of up to 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 ϵ-prefix free-encoding. I stress here that the mapping or 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 inputs a, whose encoding will be the same under the key k. But the security guarantee that we obtain from this ϵ-prefix free-encoding is that even though there could be several such candidates 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 ϵ. 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 inputs (a, b), such that encoded a is a prefix of encoded b or vice versa, should be upper bounded by the probability ϵ. And if you ensure that ϵ is very small, then we are done.

(Refer Slide Time: 14:30)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Randomized Prefix-free Encoding

❑ Randomized ϵ-prefix-free encoding for block-wise F_{CBC} :

$k \in_R \{0,1\}^n$
$a = (a_1, a_2, \dots, a_u)$
$rpf(k, a) = (a_1, a_2, \dots, a_u \oplus k)$

$rpf : \{0,1\}^n \times \{\{0,1\}^n\}^{\leq \ell} \Rightarrow \{\{0,1\}^n\}^{\leq \ell}$

❑ Claim: The mapping rpf is a $(^1/_{2^n})$–prefix–free encoding

- Let $a, b \in \{\{0,1\}^n\}^{\leq \ell}$, where $a = (a_1, \dots, a_u)$, $b = (b_1, \dots, b_v)$ and $a \neq b$
  - If $u = v$, then for all choices of k, we have $rpf(k, a) \nsim rpf(k, b)$
  - If $u < v$ then $rpf(k, a) \sim rpf(k, b)$, only if $a_u \oplus k = b_u$ --- probability $^1/_{2^n}$

$k = a_u \oplus b_v$

So what we are going to now do is we are going to define a randomized ϵ-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, say u number of blocks, where u is upper bounded by ℓ and each block is of size n bits. And we have a key k with which we want to be encoding.

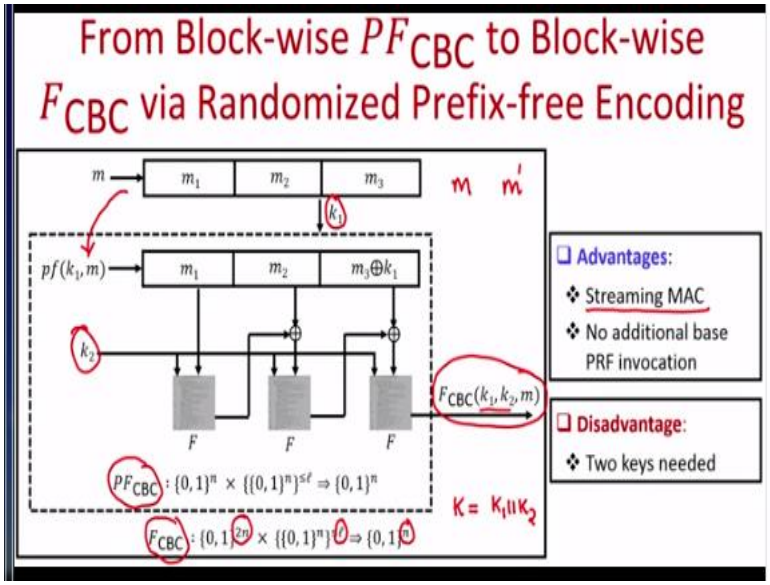
So the encoded output a will be as follows: you repeat all the blocks of a except the last block. And the last block is basically the XOR of the last block of the actual a and the key k, so that is how we are actually computing the encoded a. And my claim is that if the key for this encoding is selected uniformly randomly from the set of n-bit strings, then this mapping constitutes a $\frac{1}{2^n}$-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 $\frac{1}{2^n}$, which is definitely a negligible function in the security parameter. So let us prove that.

So imagine you have an input a, a block sequence a and another input b, a sequence of blocks and say a and b are different. And we want to find out what are the chances that for 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 the 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 b except the last block will be present in the respected encoded outputs. And the last block will be the XOR with respect to the key, so definitely all of them will be different and 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 u in a is less than the number of blocks v 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 $a_u \oplus k = b_u$ holds. 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 u^{th} block of b. And since the key for the encoding is chosen randomly from the set of n-bit strings, the probability that indeed key satisfies this relationship is $\frac{1}{2^n}$.

That means, the coding that we have seen is a very simple encoding, but it has a very powerful property that it gives you a $\frac{1}{2^n}$-prefix-free-encoding. Namely if 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, b), such that either the encoded a is a prefix of encoded b or vice versa.

(Refer Slide Time: 18:33)



From Block-wise PF_{CBC} to Block-wise F_{CBC} via Randomized Prefix-free Encoding

$PF_{\text{CBC}} : \{0,1\}^n \times \{\{0,1\}^n\}^{\leq \ell} \Rightarrow \{0,1\}^n$

$F_{\text{CBC}} : \{0,1\}^{2n} \times \{\{0,1\}^n\}^{\ell} \Rightarrow \{0,1\}^n$

❑ Advantages:

- Streaming MAC
- No additional base PRF invocation

❑ Disadvantage:

- Two keys needed

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 as we used for the case of deterministic encoding. 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 n bits.

So that is why the overall key for the full block-wise fully secure CBC will be of size $2n$ bits and it can support a sequence of blocks consisting of up to ℓ blocks, where each block is of size 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 that the overall key for the CBC PRF is interpreted as 2 chunks of n-bit keys. So the first part is 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-encoding. And the size of the encoded output and the size of the input remains the same, that is important. 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.

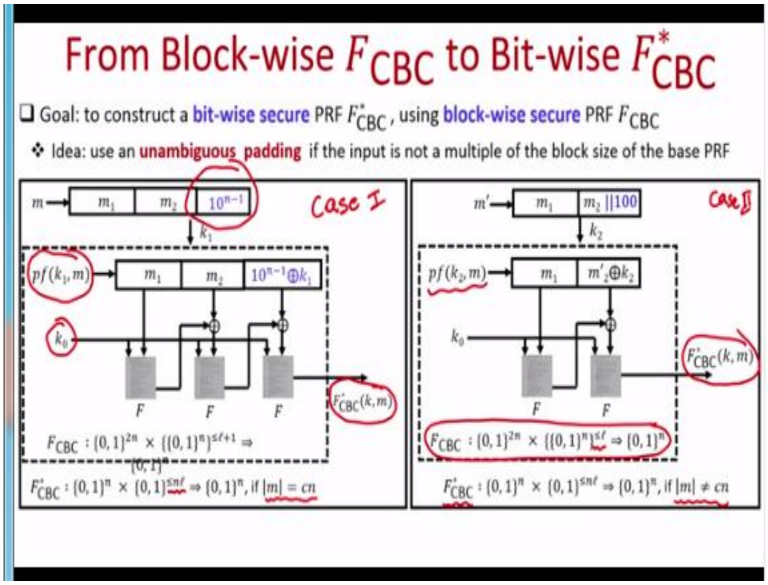
And once we have the encoded input, we now apply the block-wise prefix free secure PRF with the second part of the key. So that is why we need 2 keys, one for doing the randomized encoding and one for doing the actual PRF computation. And whatever output we obtained from the block wise, prefix-free secure 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', such that the encoded m is a prefix of encoded m' or vice versa, 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. 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 here is that you now need to operate with 2 keys, whereas if we see the deterministic prefix free-encoding, we were operating only with one key, so now you have a trade-off.

(Refer Slide Time: 22:06)



From Block-wise F_{CBC} to Bit-wise F^*_{CBC}

❑ Goal: to construct a bit-wise secure PRF F^*_{CBC}, using block-wise secure PRF F_{CBC}

❖ Idea: use an unambiguous padding if the input is not a multiple of the block size of the base PRF

m
m_1
m_2
10^{n-1}
Case I
k_1
$pf(k_1, m)$
m_1
m_2
$10^{n-1} \oplus k_1$
k_0
F
F
F
$F^*_{\text{CBC}}(k, m)$
$F_{\text{CBC}} : \{0,1\}^{2n} \times \{\{0,1\}^n\}^{\leq \ell+1} \Rightarrow \{0,1\}^n$
$F^*_{\text{CBC}} : \{0,1\}^n \times \{0,1\}^{\leq n\ell} \Rightarrow \{0,1\}^n$, if $|m| = cn$
m'
m_1
$m_2 \| 100$
Case II
k_2
$pf(k_2, m)$
m_1
$m'_2 \oplus k_2$
k_0
F
F
$F^*_{\text{CBC}}(k, m)$
$F_{\text{CBC}} : \{0,1\}^{2n} \times \{\{0,1\}^n\}^{\leq \ell} \Rightarrow \{0,1\}^n$
$F^*_{\text{CBC}} : \{0,1\}^n \times \{0,1\}^{\leq n\ell} \Rightarrow \{0,1\}^n$, if $|m| \neq cn$

So now let us see how we go about 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 bits 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 n bits. Now you want to design a PRF which I denote as F^*_{CBC}, taking a key of size n bits, and it can take any input bit sequence of length up to $n\ell$ bits. And now there could be 2 possible cases, depending upon whether the input for this PRF F^*_{CBC} it's size is equal to some multiple of n or not. So remember I am assuming that the block size of my base PRF is n bits.

So case 1 could be when the input size of m is some constant c times n. And case 2 could be when the size of the input is not equal to any constant time n, that means it is not a multiple of n. So depending upon 2 cases, we operate the so called PRF F^*_{CBC} 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 $2n$ 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^*_{CBC} 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^*_{CBC} 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_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 $\ell + 1$ such blocks. And it gives you a fixed output.

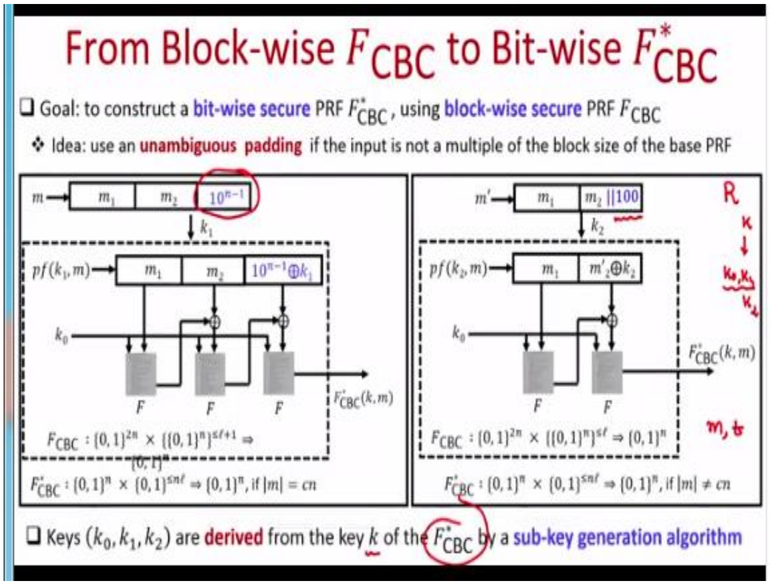
So, if you see the outer view of the PRF F^*_{CBC} that we are going to construct here, it could take an input of size $n\ell$ bits. And if my input size is already a multiple of n, that means it could have up to ℓ number of blocks. But if you see the inner invocation of the block-wise fully secure PRF then it could take up to $\ell + 1$ blocks of size little n bits. You might be wondering that why this additional 1 block? This additional 1 block might come if actually your message size is already a multiple of n, 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 $\ell + 1$ blocks. So internally what I do is I operate my block-wise fully secure PRF and for doing that actually I first take the 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 k_0, k_1 are derived. So 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^*_{CBC} 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 a 1, followed by the required number of 0's to ensure that I have the second block which also have n bits.

And then I use another sub key in this case, which I call k_2, to 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. And whatever comes as an output that is taken as the output of my PRF F^*_{CBC} for the input m on the key k, so these are the 2 cases here.

(Refer Slide Time: 28:08)



From Block-wise F_{CBC} to Bit-wise F^*_{CBC}

- ❑ Goal: to construct a bit-wise secure PRF F^*_{CBC}, using block-wise secure PRF F_{CBC}
  - ❖ Idea: use an unambiguous padding if the input is not a multiple of the block size of the base PRF

$F_{\text{CBC}} : \{0,1\}^{2n} \times (\{0,1\}^n)^{\leq \ell+1} \Rightarrow \{0,1\}^n$

$F^*_{\text{CBC}} : \{0,1\}^n \times \{0,1\}^{\leq n\ell} \Rightarrow \{0,1\}^n$, if $|m| = cn$

$F_{\text{CBC}} : \{0,1\}^{2n} \times (\{0,1\}^n)^{\leq \ell} \Rightarrow \{0,1\}^n$

$F^*_{\text{CBC}} : \{0,1\}^n \times \{0,1\}^{\leq n\ell} \Rightarrow \{0,1\}^n$, if $|m| \neq cn$

- ❑ Keys (k_0, k_1, k_2) are derived from the key k of the F^*_{CBC} by a sub-key generation algorithm

Now let us see how exactly that sub-keys k_0, k_1, k_2 are derived. So remember that the overall key for my PRF F^*_{CBC} 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^*_{CBC} 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.

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 that receiver is going to derive the sub-keys k_0, k_1, k_2.

And then it has to remove the padding because remember, receiver is to first find out what exactly was the padding that sender has performed at it's 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 makes a complete scan. And it stops 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, 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 is an indication it is in case 2.

So once 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 for the message that it has received, either by operating the PRF F^*_{CBC} with the sub-keys k_0, k_1 or with the sub-keys k_0, k_2.

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 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 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 block-wise prefix-free secure 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-free set. And then finally, we construct or convert this block-wise fully secure PRF into bit-wise secure PRF using the CBC mode. 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. I hope you enjoyed this lecture thank you.