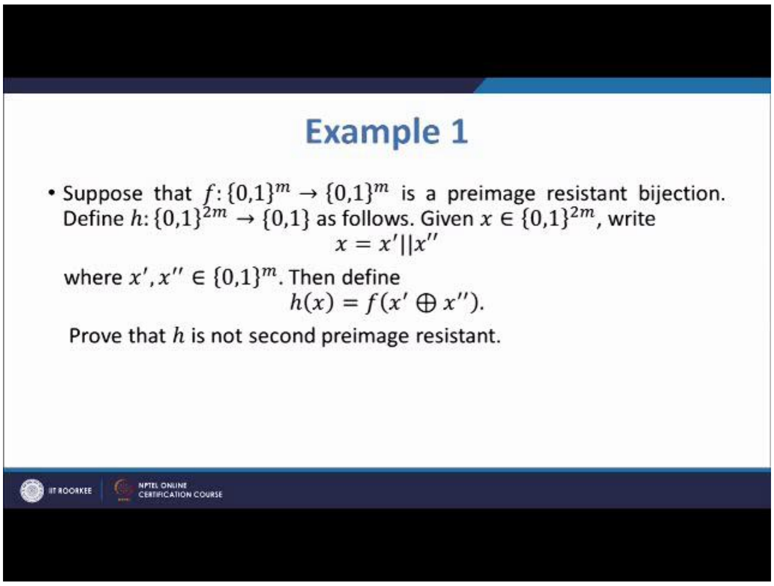
Introduction to Cryptology
Dr. Sugata Gangopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Roorkee

Lecture – 20
Problem discussion

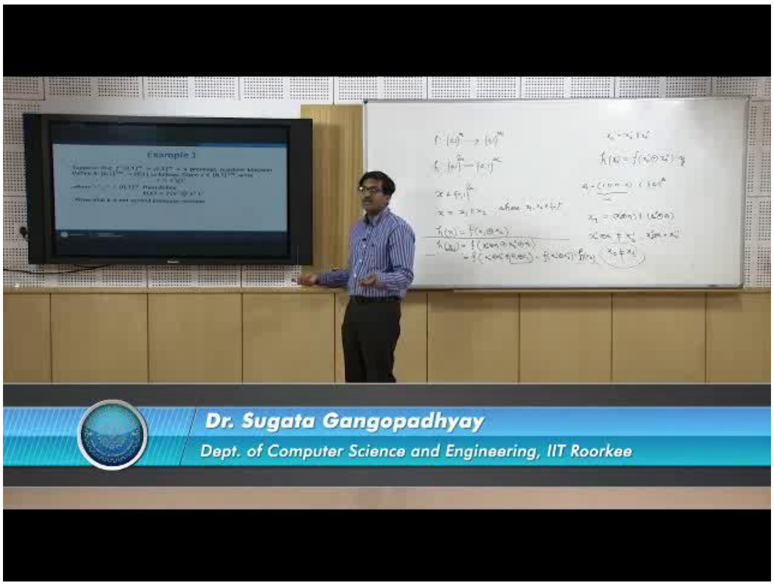
Hello. This is the last lecture of our course Introduction to Cryptology. We will be doing some problems on hash functions. I believed that you have enjoyed this course and you have also participated in the discussion forums. This lecture will be on some constructions or basically some toy constructions of hash functions and we will be checking the kind of securities they do not have or sometimes they have. So, this is Example 1.

(Refer Slide Time: 01:22)



Example 1 says that suppose we have a bijection from 0, 1 raise to the power m to 0, 1 raise to the power m which is preimage resistant. And suppose we are going for some kind of iterative construction with this bijection, the iterative construction goes this way. We are using x as a compression function and h is a function from 0, 1 to the power twice m to 0, 1 to the power. So, please read 0, 1 to the power m. Let me write over here.

(Refer Slide Time: 02:12)



I have a function which I claim to be a bijection and preimage resistant bijection from 0, 1 to the power m to 0, 1 to the power m that means, it takes m length string to m strings. And I am considering a function h from 0, 1 to the power twice m to 0, 1 to the power m, so there is a print here that will correct later. Now, how are we defining this function? Given an element x in 0, 1 to the power twice m, we realize then we can split it into left half and right half. So, we will write x as x 1 concat x 2 where x 1 belongs to 0, 1 or let us write them together x 1 and x 2 belongs to 0, 1 to the power m. Then h of x is defined as f of x 1 bitwise x or bit x 2.

The question is, well I have used in the slide x prime and x double prime here I am using x 1 and x 2 that is understandable. So now, the question is that whether this is second preimage resistant or not. Now let us look at this function. Suppose that we have got a preimage image pair, so suppose is a particular x this is equal to let us call it x 0. So, suppose this x 0 splits up into x 0 1 and x 0 2.

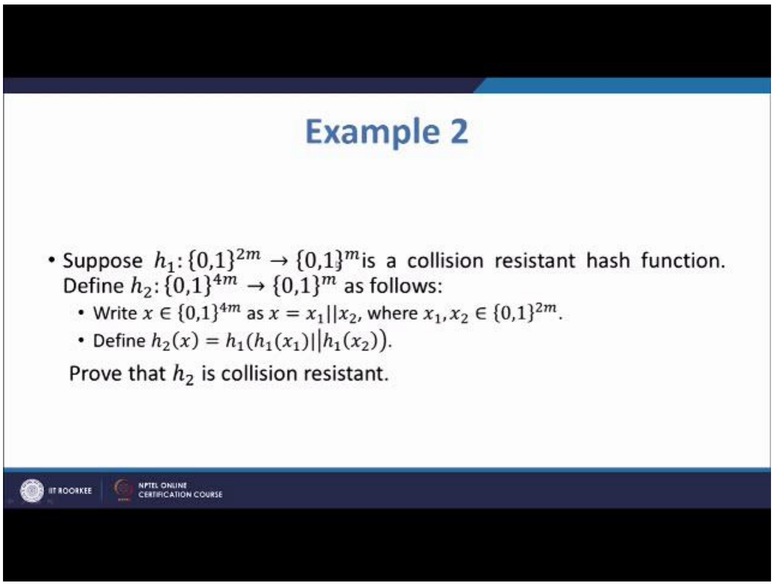
Or let me write like this here, suppose x 0 splits up as x 0 prime and x 0 double prime. I evaluate h on x 0 to obtain f x 0 prime XOR f x 0 double prime and I claim that I have this value. Now I have to find an x 1 which is not equal to x 0 and whose evaluation is this. This is not difficult because, suppose we take a string let us say 1 and all 0's, all together m bits belonging to 0, 1 raise to the power m. Let us call this e 1.

Now let us construct a function like this sorry, let us a construct a point x 1 which is this x 0 XOR e 1 concat x 0 prime or e 1 and x 0 double prime XOR e 1. Now one thing is sure that x 0 prime plus e 1 is not equal to x 0 prime, because if it happened then that would have meant e 1 equal to all 0 strings which is not so. It is also clear that x 0 double prime plus e 1 is not equal to x 0 double prime the argument is again same.

Therefore, we can say that x 0 is equal to x 1, but if we evaluate the hash function at x 1 let us see what happens. H x 1 equal to h x 0. We know that this is the left and this is the right half so we can directly write f of x 0 prime plus e 1 plus x 0 double prime plus e 1, and therefore f of x 0 prime plus x 0 double prime plus that is XOR e 1 that is XOR e 1. Now we know that e 1 XOR e 1 bitwise is going to be 0 therefore I arrive at f of x 0 prime plus x 0 x 0 double prime which is equal to h of x 0. Thus, we see that we have obtained another point x 1 such that h of x 1 is equal to h of x 0, but x 1 is not equal to x 0. Thus, we have obtained a second preimage.

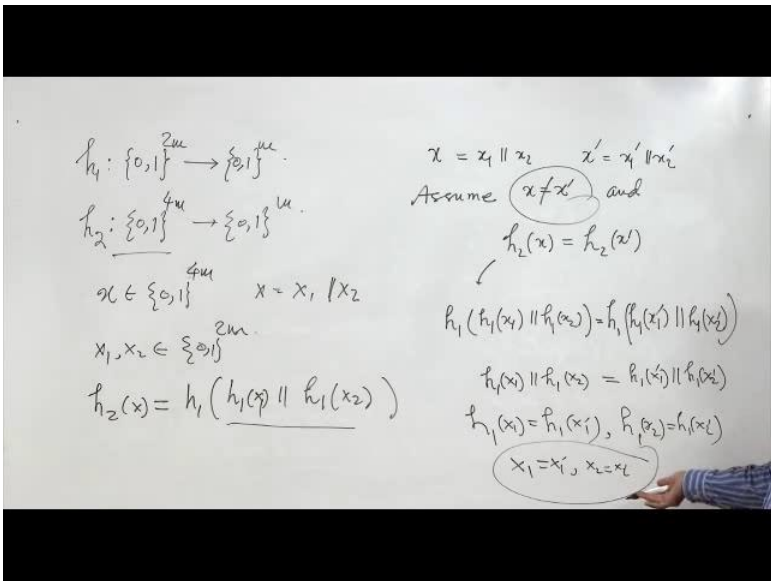
So, this is why this construction leads to hash function which is not second preimage resistant all though the compression function used is preimage resistantheer.

(Refer Slide Time: 10:41)



Here we reclaim that we have a collision resistant hash function h 1 which is a compression function in fact and which is from 0, 1 raise to the power 2 m to 0, 1 m.

(Refer Slide Time: 10:51)



So, let me write it over here h 1 is a function from 0, 1 raise to the power 2 m to 0, 1 raise to the power m is a collision resistant hash function or compression function. And we have a scheme here which says that here defining h 2 from 0 raise to the power 4 m to 0, 1 raise to the power m. As follows I take an element of x belonging 0, 1 raise to the power 4 m I split up into two parts each part belonging to 0, 1 to the power 2 m. I am taking x inside 0, 1, rise to the power 4 m and split it up.

So I am splitting up x into x 1 plus x 1 concat x 2, where x 1 and x 2 both are elements of 0, 1 raise to the power 2 m and then I use the original concat function, sorry original concat function. So, I get h 2 x is equal to h 1 of h 1 x 1 concat h 1 x 2 so this is my definition.Of course, h 1 can accept 2 m bit string and here also h 1 can accept a 2 m bit strings and h 1 maps any 2 m bits strings to m bit strings so I get m bit string over here, I get a m bit string over here joining them I have a 2 m bit string. So, again h 1 can accept this I apply h 1 on it and I get the value of h 2. So, that is my construction.

Now I have asked to prove that h 2 is collision resistant. The question is how to prove this? What we do is that we in the beginning assumed that it is not collision resistant, suppose if possible h 2 is not collision resistant. That means that I should be able to pair an elements of 0, 1 to the power m which are the same image.

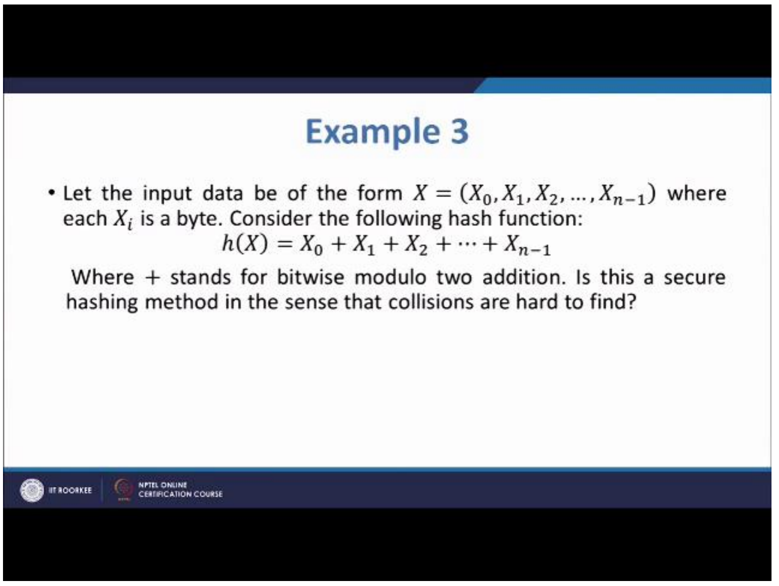
Now let us denote this pair by symbols. So we have got a point is a x and x splits up as x 1, suppose this is one number, one point and x prime which splits as x 1 prime concat x 2 prime this is another string. And I am claiming that they are not same, x is not equal to x prime that is my claim have assume. Such that I assume that this gives a raise to a collision, so h 2 x is equal to h 2 x prime suppose that it happens.

Now let us try to see, what are the logical consequences of this assumption? If this is true that if I am able to find out collision in h 2 then from this I will have h 1 h 1 x 1 concatenation h 1 x 2 is equal to h 1 h 1 x 1 prime concatenation h 1 x 2 prime is my definition. Now one thing is clear to me that by my initial claim h 1 is collision resistant. Therefore, if h 2 is not collision resistant we arrive at an equation like this, and since h 1 is collision resistant then I cannot get a collision but this hash values are equal. That means, hash values are equal means this forces me to say that h 1 x 1 concatenation h 1 x 2 is equal to h 1 x 1 prime concatenation h 1 concatenation x 2 prime. I am forced to say this suppose it is not so that means I have obtained a collision of h 1 which is not possible

And therefore, since these two strings are equal then h 1 x 1 is equal to h 1 x 1 prime and h 1 x 2 is equal to h 1 x 2 prime. Well, again we know that h 1 collision resistant. So, when I am got an equation like this it forces me to say that x 1 is equal to x 1 prime and x 2 is equal to x 2 prime. That means, that this assumption contradicts this, but I am forced to say this because my basic claim is that h 1 is collision resistant. And therefore, I cannot assume that h 2 is not collision resistant, h 2 is also collision resistant.

So, what we see in this small problem is essentially what we studied in iterative construction and Merkle Damgard construction. In Merkle Damgard construction is more complicated and it is quite elaborate, but it achieves ultimately the same thing. It says that if you assume that the compression function used in Merkle Damgard construction is collision resistant then whatever you are going after iteration by using that rule is going to be collision resistant. Of course, this is not Merkle Damgard construction but this is another construction, but it shows the same kind of property.

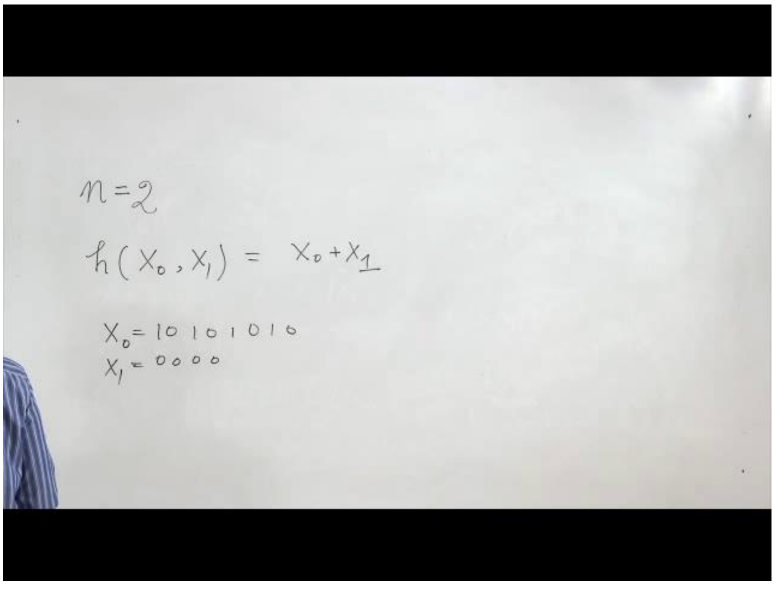
(Refer Slide Time: 19:33)



Let us now move to the last problem of this session. This is a very easy problem will do this and close this series of lectures. For the time being it is a very easy problem. It just says that suppose that we have input data in the form capital x is equal to x 0 x 1 x 2 and to x n minus 1 such that x i are bytes.

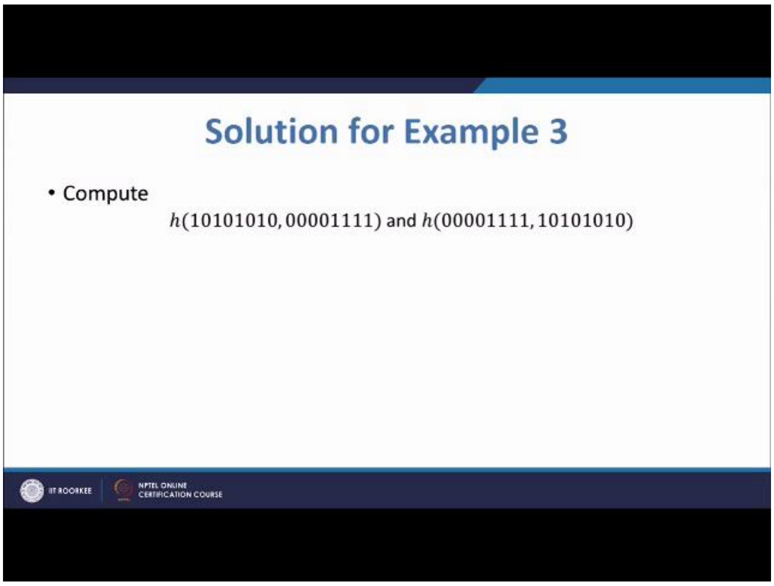
These are 8 bit segment. And suppose my hash function is just summing up all these things and let us assume that this sum is bitwise addition modular 2 bitwise XOR. And so the question is this secure, the answer is it is not secure.

(Refer Slide Time: 20:46)



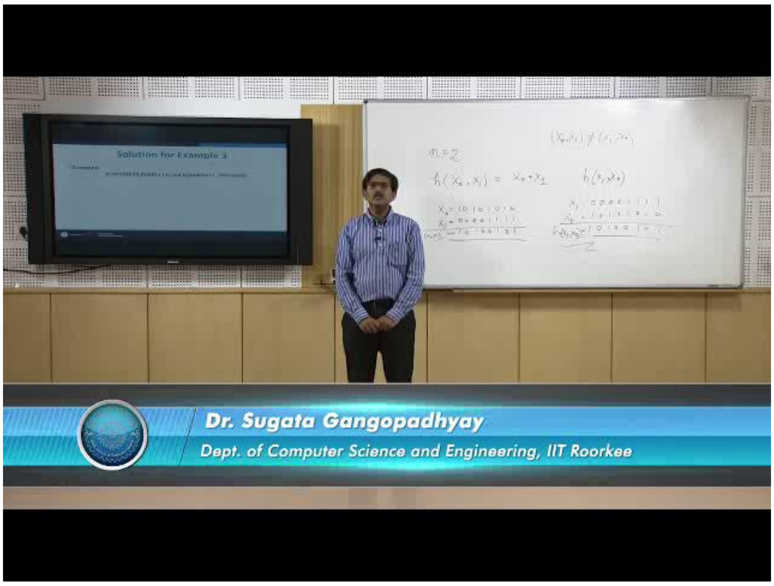
So, we will do it just by taking n equal to 2. If we have n equal to 2 and then the hash function is like let us say x 0 comma x 1 is sum x 0 x 1. So let us look at this as a closing remark, so we have got 2 bytes; one byte is 1 0 1 0 1 0.

(Refer Slide Time: 21:40)



So, suppose I take x 0 equal to 1 0 1 0 1 0 and x 1 0 0 0 0 another 1 0 and x 1 is 0 0 0 0 1 1 1 1 add up 1 0 1 0 0 1 0 1.

(Refer Slide Time: 21:47)



So, my hash value is this. Now suppose I inverted the order. Suppose, I evaluated this over x 1 x 0 and of course this sequence x 0 x 1 is not equal to x 1 x 0 and then I also get the same result, because this addition is commutative 0 0 0 0 1 1 1 1 and x 0 is 1 0 1 0 1 0 1 0 so if I add up I will get 1 0 1 0 0 1 0 1, the same. And therefore we have obtained a collision, and therefore this is not a secure hash function.

By this I end today's lecture and this is the last lecture of our course. I hope that you have enjoyed the course and I wish you all the best for the final examination.

Thank you very much.