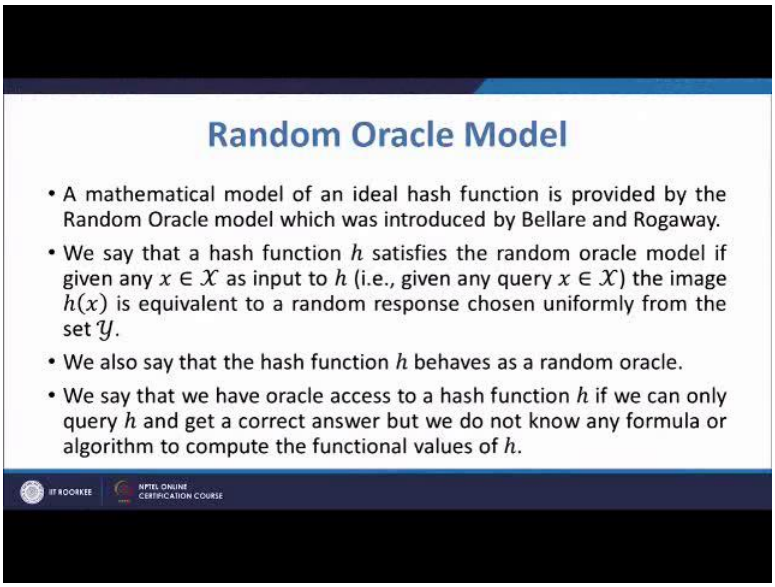
Introduction to Cryptology
Dr. Sugata Gangopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Roorkee

Lecture – 17
Random Oracle Model, Security of Hash Functions

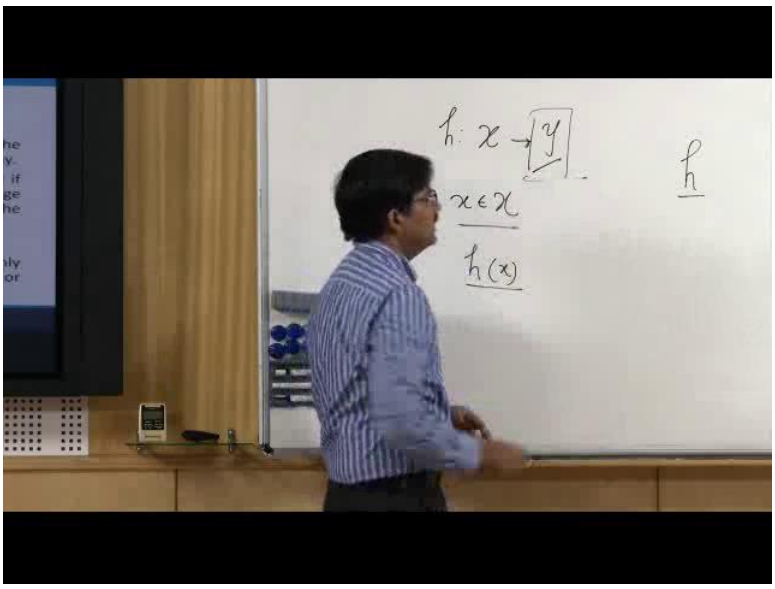
Hello, welcome to week 4, Lecture - 2. We have been discussing cryptographic hash functions, and we continue our discussions on cryptographic hash functions.

(Refer Slide Time: 00:24)



So, in this lecture, we will start with discussing random oracle model, a mathematical model of an ideal hash function is provided by the Random Oracle model which was introduced by Bellare and Rogaway. Now, we say that a hash function h satisfies the random oracle model if given any x belonging to capital X as input to h that is given any query x coming from the set X - capital X the image h x is equivalent to a random response chosen uniformly from the set y. Now let us try to understand this briefly.

(Refer Slide Time: 01:32)



So, it basically means that the function which we are using as a hash function should have some kind of random behavior. It means that when I am given an input x to hash function, the output will be somewhat indistinguishable from a random function, a random value from the set Y; particularly, we assume that we do not have any knowledge about the formula or an algorithm by which h is computing the image value h(x). So, this is our assumption. We also say that the hash function h behaves as a random oracle or sometimes we say that we have oracle access to hash function if we can only query h to get a correct answer, but we do not know any formula or algorithm to compute the functional value of h. So, this is random oracle model.

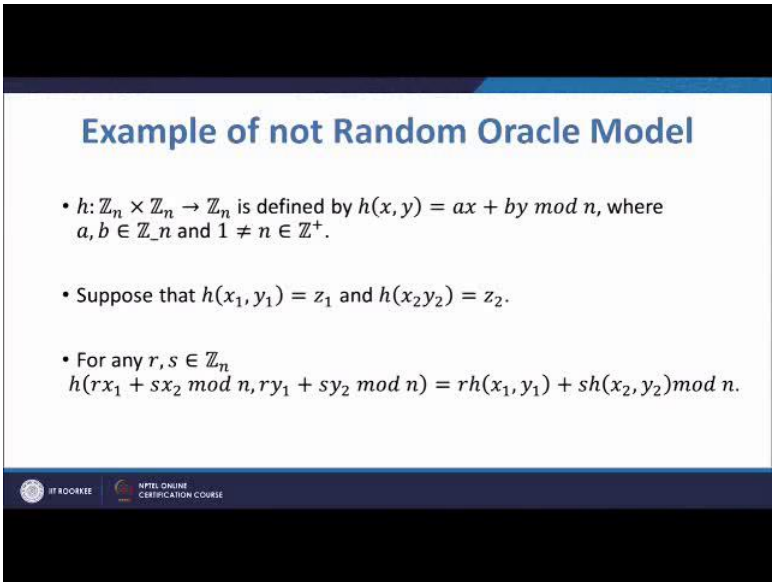
So, to sum up, we will assume that we have a hash function h, and we do not know or we do not have any access to a formula, which determines the values of h or an algorithm, so h is some kind of a black box. Now what we can do is that we can query h, and that is the context in which say that h behaves like an oracle that is we can query h and h will give me back the answer and that answer will be correct.

But if I keep on querying h, I will not be able to get any pattern, and no matter how many times I have queried h before, the next query will be will behave randomly that is a next the answer to the next query will seem to be coming from the space y equiprobably. And if we assume this property on h then we are following the random oracle model. And we will

discuss the problem of pre image, second pre image and collision with respect to this random oracle model.

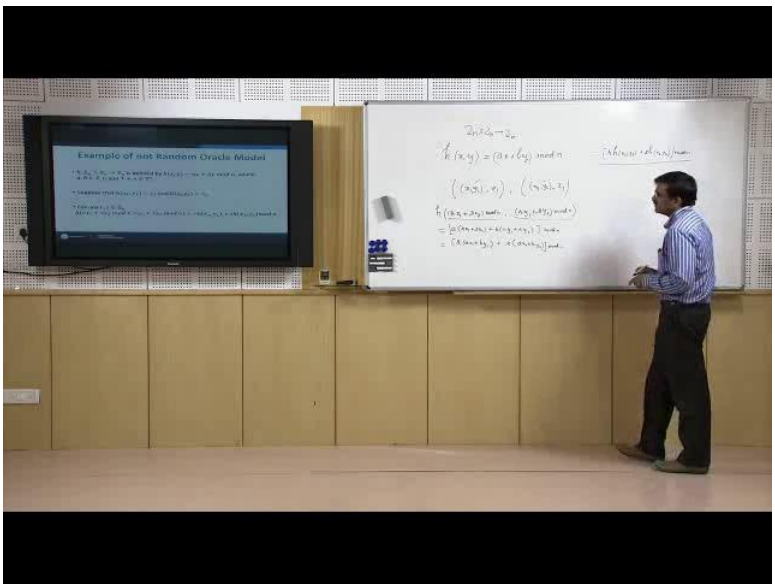
Now, the point here is that if we assume our hash function to be following random oracle model then we are not assuming any extra information on the hash function. So, if a hash function for certain a parameter value is insecure in the random oracle model, we can very reasonably say that it is going to be insecure, because if it is not a random oracle model, then we will know some extra information about h. So, that is the kind of that is the sense in which we are looking at this subject.

(Refer Slide Time: 05:18)



Next, we check an example of a hash function, which is not following the random oracle model. Now let us look at this we have got the set Z by Z sub n that is a set of all integers modulo n cross Z sub n going to Z sub n. So, I have a function from the Cartesian product of Z sub n by itself to Z sub n. And I am defining the function in this way, it is operating on an ordered payer h (x, y) and there are two fixed values a and b.

(Refer Slide Time: 06:21)



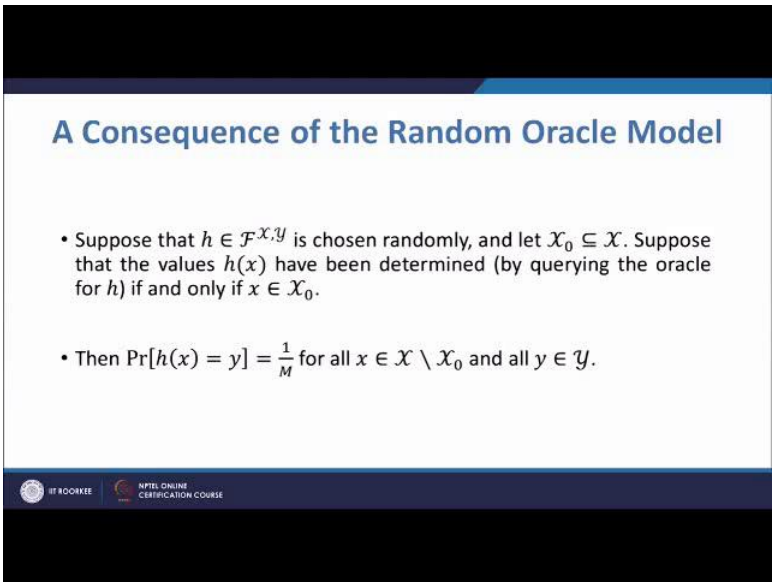
And I am computing h (x, y) as a x plus b y modulo n. Well, this is a function from Z sub n to Z sub n. It is quite reasonable to say that well this is a comprehension function, because it is mapping a large larger space onto a smaller space. Now, suppose we have got two pairs of image and pre image. So, essentially what we are having is that we have coherent this hash function twice and we are getting results like this. So, one result is x 1 y 1 comma Z sub 1, and another pair is x 2 y 2 comma Z sub 2, so this what we have. Now, with this information, we are trying to determine the value of h at other positions without computing the formula a x plus b y.

So, what we what we can do what we see is that if we take any r, s in Z n, and then create to payers r x 1 plus s x 2 mod n and r y 1 plus s y 2 mod n. And let us see what happens h of r x 1 plus s x 2 mod n comma r y 1 plus s y 2 mod n is equal to a times r x 1 plus s x 2 plus b times r y 1 plus s y 2, this whole mod n. Now, this is equal to r times a x 1 plus b y 1 plus s times a x 2 plus b y 2 this whole thing mod n, and therefore we will get this is equal to r times h (x 1, y 1) plus s times h (x 2, y 2) this whole mod n.

Now, what do we have after doing this? Now this means that if I have this pair, then this function is such that without evaluating the function, I will be able to know the values of the function at several points, namely points of this type. And therefore, if I later on query at points of query those points, whatever answer I will get of the hash value and not equally

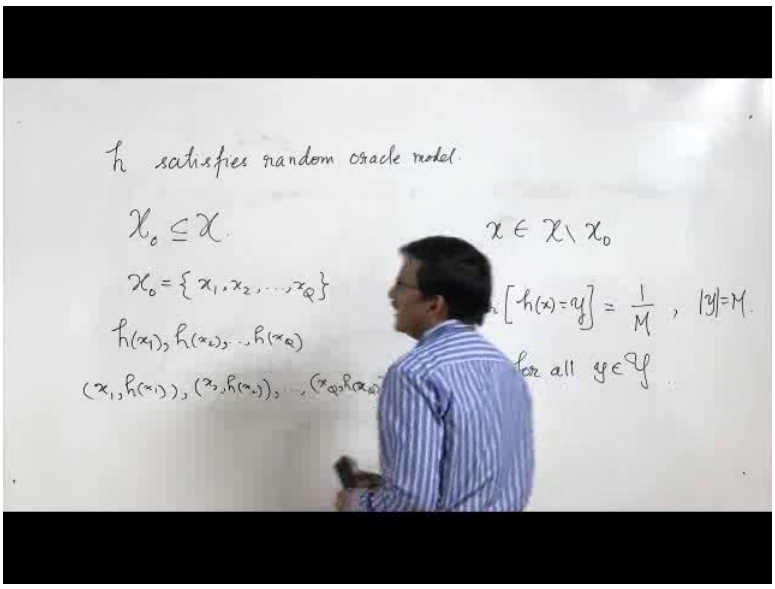
distributed what the co domain. It is determines with probability one, we will get these values. And therefore, this is not this function does not satisfy the random oracle model.

(Refer Slide Time: 11:19)



As a consequence of this model, we have a result or a very basic, yes a very basic result as a consequence of this model, and which says that suppose h is a function from x y which is chosen randomly and let x 0 is a subset of X suppose that the value h (x) have been values h (x) have been determined by querying the oracle of for h if and only if x is in X 0 if that happens then probability that h x is equal to y is 1 by M for all x belonging to x set minus X 0, for all y belonging to Y.

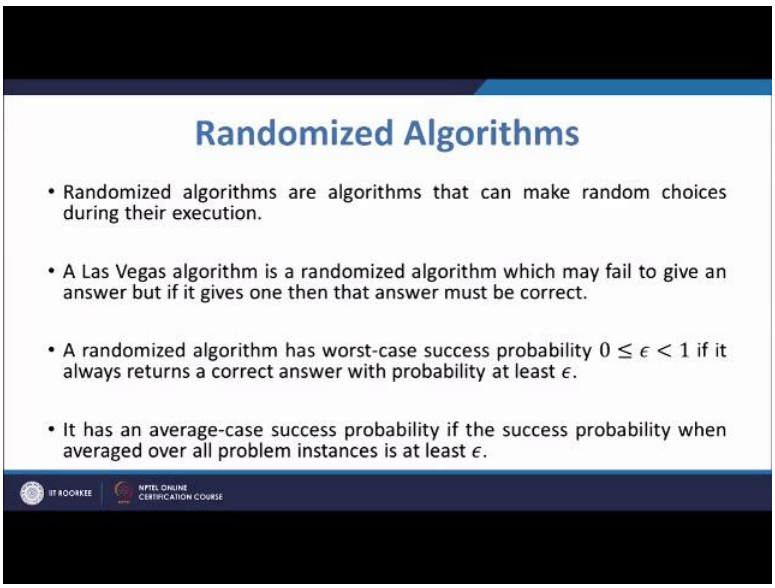
(Refer Slide Time: 12:10)



Now, let us try to understand these results. Basically, this result says that suppose h satisfies random oracle model, and suppose that we have picked up certain points from x and the set of that those points is X 0, which is a subset of x. And suppose we have queried the oracle for h at those points, so it might be like this that X 0 is something like this x 1 x 2 and so on up to x Q, where the Q is the number of queries that we have made, and we are getting the hash values like this h (x, Q).

So, we have got several accesses to several valid pairs like this. We have got access to several valid pairs like this. Now if x has random oracle model then if I take any x outside X 0 then even after having the knowledge of these valid pairs the probability that h (x) equal to a certain y is 1 by M, where M is the number of elements in y for all y belonging to capital Y, that is to say the hash function does not leak any information. So, we will be studying the problems of pre image, second pre image and collision by using this result.

(Refer Slide Time: 15:08)

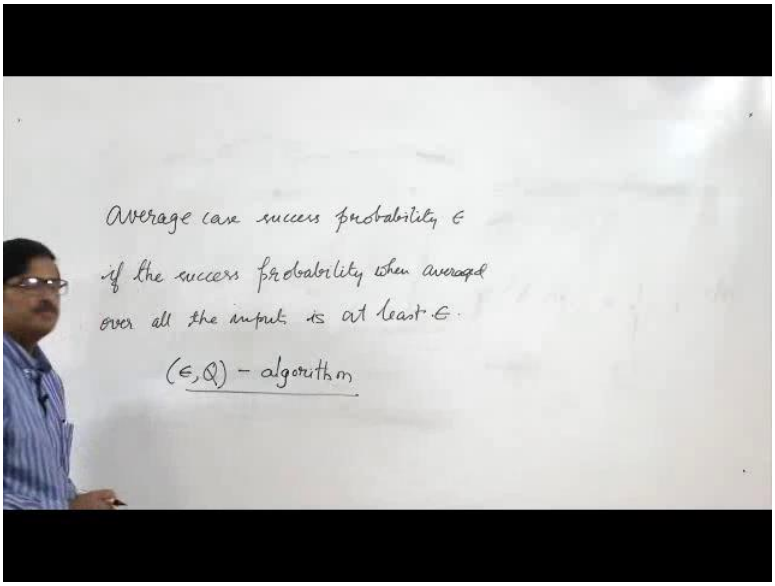


Now, we come to another concept, which is called randomized algorithm. Now here randomized algorithms are algorithms that can be that can make random choices during their execution. We have already seen randomized algorithms while checking the primality testing algorithms; there we saw that at some point, the algorithm is choosing a number at random from certain range. So, here also these algorithms will be making random choices, but the randomized algorithms that we saw for primality testing are somewhat different from the randomized algorithms that we will be discussing in these lectures.

So, these algorithms are called Las Vegas algorithms, which are such that if they return an answer that answer will be correct, otherwise they will not return an answer. The other algorithms the primality testing algorithms has such that they will always return an answer and the answers might be wrong from time to time, but here these are different algorithms; these are Las Vegas algorithms.

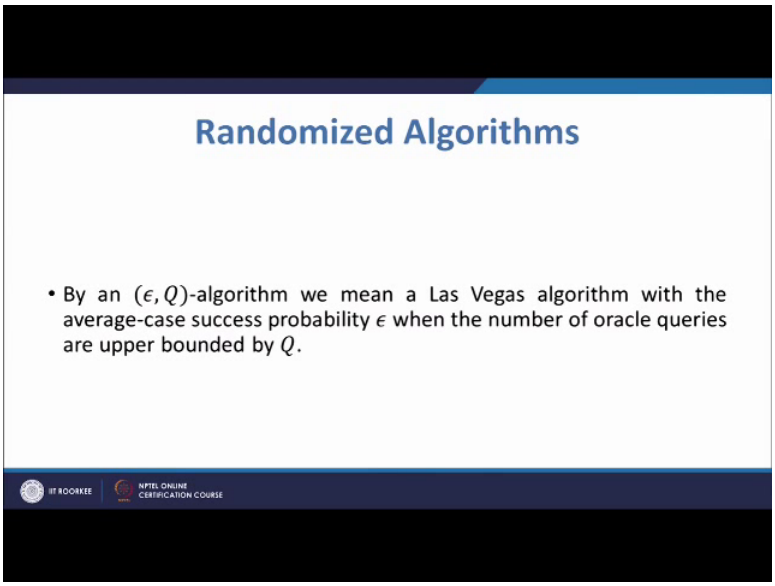
So, a Las Vegas algorithm is randomizes algorithm which fail to give an answer which may fail to give an answer, but if it gives 1 then the answer must be correct. A randomizes algorithm has worst case success probability epsilon line between 0 and 1, if it always returns a correct answer with probability at least epsilon. It has an average-case probability success probability epsilon if it has an average case success probability if the success probability when averaged over all problem instances is at least epsilon.

(Refer Slide Time: 17:26)



So, I will make this last point clear that we will say that there will be an epsilon over here. So, we will say that an algorithm we will have as an average-case success probability epsilon, if the success probability when averaged over all the input is at least epsilon; of course, epsilon is between 0 and 1.

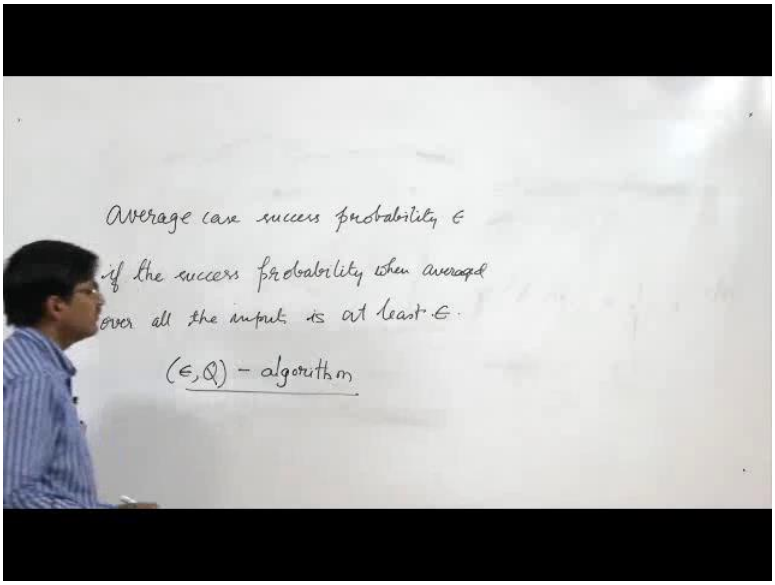
(Refer Slide Time: 18:33)



Now, we have another terminology. An epsilon Q algorithm is a Las Vegas algorithm with the average-case success probability epsilon when the numbers of oracle queries are upper

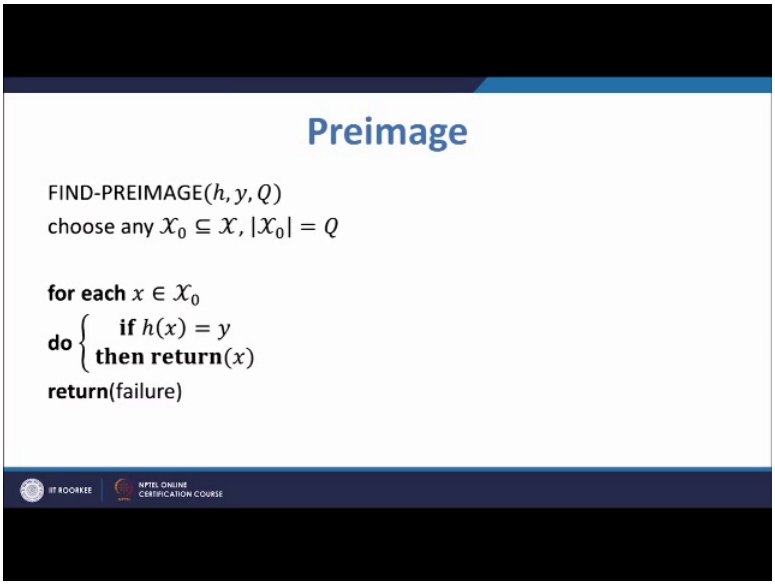
bounded by Q.

(Refer Slide Time: 18:51)



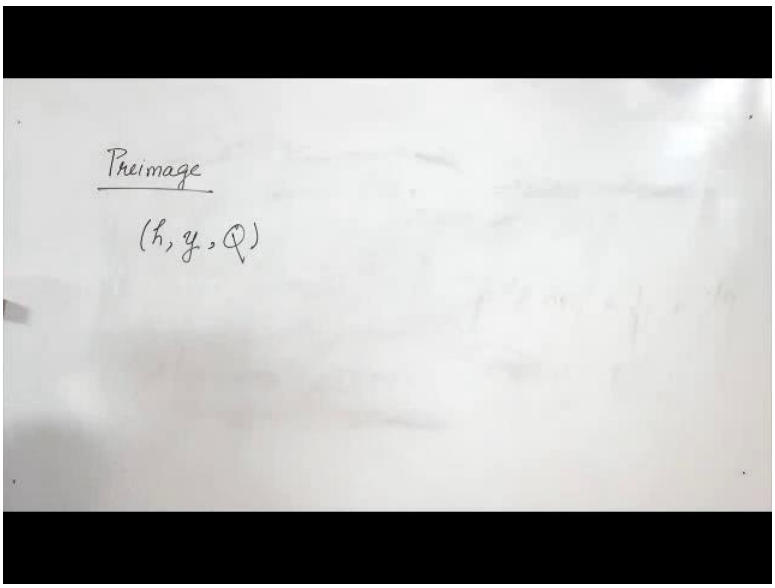
So, we will call an algorithm an epsilon capital Q algorithm if its average-case success probability is epsilon when we are allowed to make utmost Q queries.

(Refer Slide Time: 19:33)



Now, we come to the problem preimage. We know what is the problem preimage; we will recall that again.

(Refer Slide Time: 19:53)

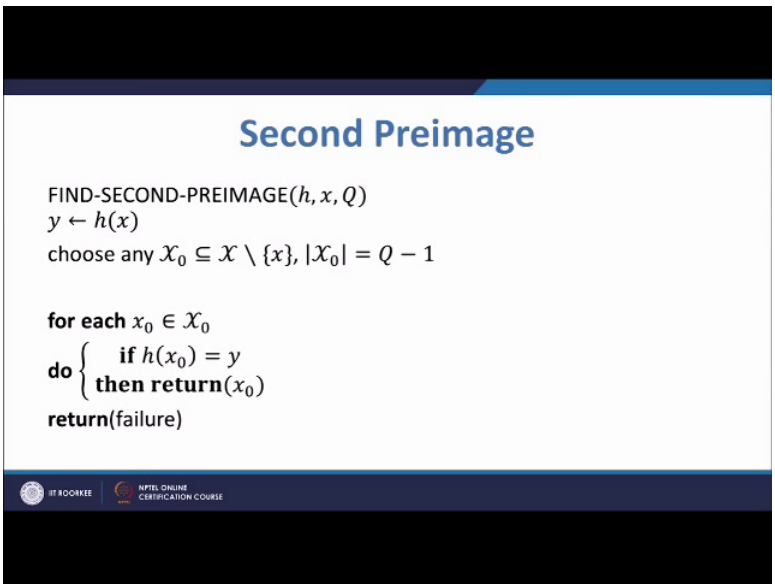


Preimage - now we are given a hash function an image, and a upper bound of number of queries that we can make and we are asking a question that how to determine the preimage of y. The steps are as follows. We choose a set of queries that is X 0 at random and taking care that the number of queries that is the cardinality of X 0 is Q, and then we keep on taking elements of X 0 and asking whether h(x) is equal to y. Where y is the input of the image y is the image that is that comes as the input.

So, if we are successful then we will return x, we will say that we have success, but otherwise we will say it is a failure. So, see that it is a Las Vegas algorithm because it is quite possible that we will not be able to get the preimage, because we have chosen some points. But if there is one point x for which h (x) equal to y when you are hundred percent sure that this is the pre image of y. So, this is a Las Vegas algorithm for preimage.

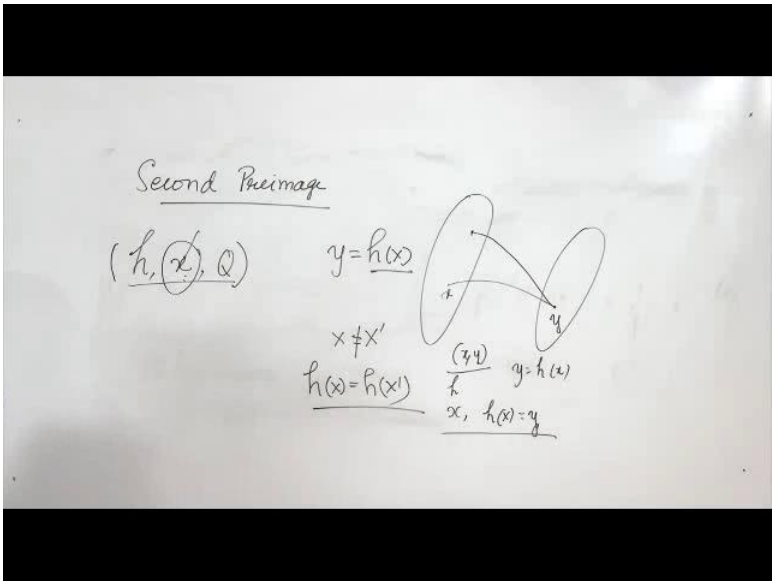
Now, let us go to second preimage.

(Refer Slide Time: 21:46)



Let us recall the problem of second preimage. Now the problem is like this that we have got an input as a hash function h and a value a data x and a number of queries this comes as the input.

(Refer Slide Time: 22:15)



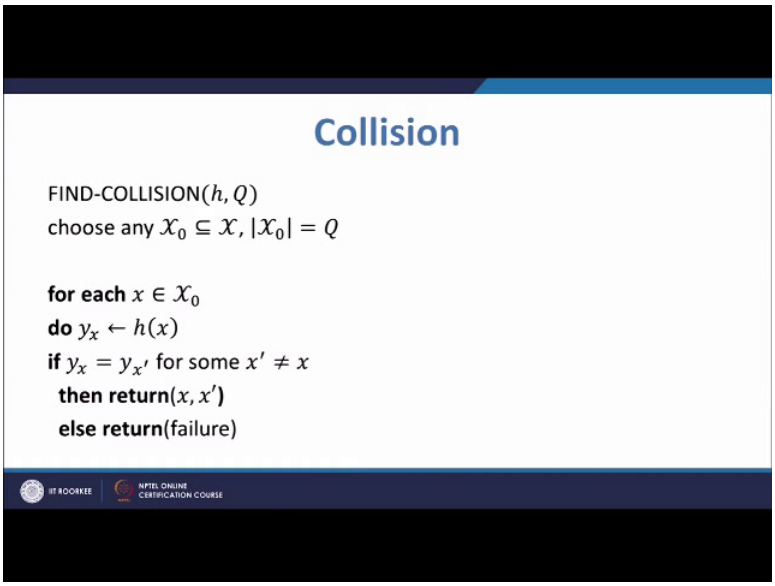
Now one may ask me that you are talking about second preimage then why are you giving a input value rather than an output of value of h. The answer is like this. Let us look at the diagram. So, often it may seem that when I am asking somebody to find out second preimage

that means, I am giving him an image and a preimage and asking him to find another preimage which maps to the same y. Now what we will say that this is giving a pair x comma y is same as of course, such that y is equal to such as h (x) is same as giving the input x and the hash function h, one can always determine h x which is equal to y.

So, when we talk about the second preimage problem, we say that the input is h, x, Q, where x is the input to the hash function, and we can always find out one output value that is one hash value that is y. And our goal is to find another element X prime which is not equal to x such that x is equal to h (x) is equal to h (X prime), this is our goal. And now let us look at the algorithm, it is only few steps and it is very much like the preimage algorithm. Here we have we changed the value changed the symbol here, we are saying that x 0 is a genial element of capital X 0, because we have already used x over here

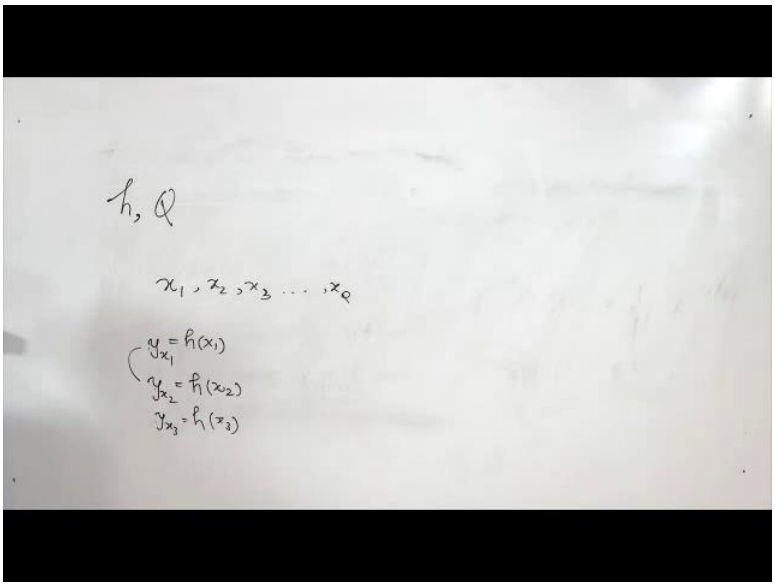
So, first we are choosing a set of cardinality Q that is X 0, and we know and that contents x because anyway we have to query the oracle for x and get the image the first image. Now so we will take out x from capital X and consider a subset of that x 0. So, the cardinality of X 0 is Q minus 1; and after that we start querying h by using the elements of X 0. If we hit up on something then we assure that we have got a second preimage because x is not inside X 0. And if you do not hit up on it then we say that we have a failure. So, this is a Las Vegas algorithm to solve the problem second preimage.

(Refer Slide Time: 25:38)



Lastly, we come to the problem collision. Now let us look at this problem. So, here an algorithm to solve collision has input h and Q.

(Refer Slide Time: 25:56)



So, we have h the hash function, and Q is the number of queries allowed to us. Then what we do we again choose some X 0, a set x 0 of inputs. And then for each x in X 0, we start evaluating h or querying the oracle for x and we store the values in y x. And at each step, we ask whether we have hit up on something a y x, which has already occurred. So, it goes like this, we have x 1, x 2, x 3 and so on up to x capital Q. So, we start like this. We compute y x 1 which is equal to h (x 1); and of course, that y x 1 is something new then because we have not we have not queried h before.

After that we query again with x 2 whatever we get we write at y x 2. Then we ask a question at this step whether they are same or not; if they are same then we have got a collision; if they are not same we precede y x 3 which is h of x 3 we precede. We again ask whether this is equal to this or this if it happens then we have a collision otherwise we proceed; we proceed Q steps. In between if we have a collision we say that we have found a collision otherwise not, otherwise we have a failure.

In the next lecture, we will study these algorithms more closely, and find out their probability values that is success probabilities of each of these algorithms. For the time being, this is the end of today's lecture.