Internetwork Security
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

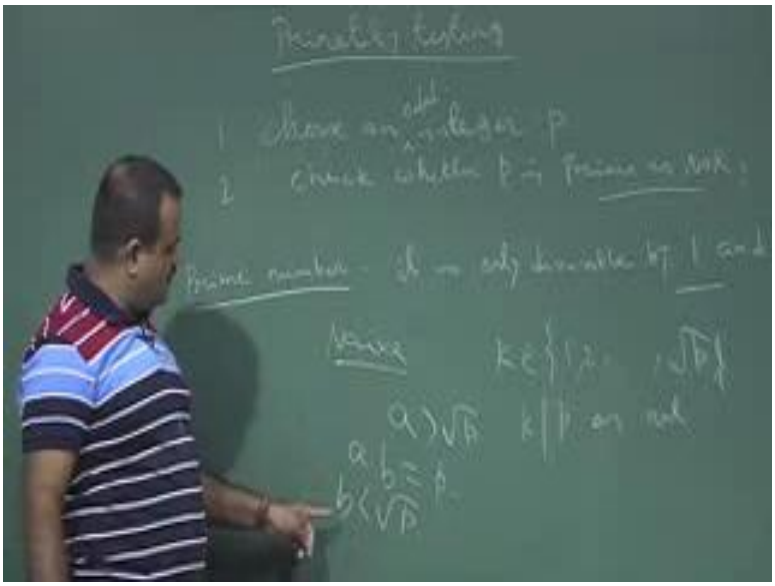
Lecture - 27
Primarily Testing

So, we talk about primarily testing of a number we need a prime number, so for our public key cryptosystem. So, you want to see whether we can get a prime number say suppose we need for RSA, suppose we need 1 0 we need p, q large prime say 1 0 2 bits. So, we need a number prime which is lies between 0 less than p, q less than 2 to the power this because this many bits actually, so huge number.

(Refer Slide Time: 00:40)

So, now, the question is, is there any algorithm which can give us this many bits prime number. So, is there any algorithm in which we can generate a weak is this is the prime number algorithm say we will give an input say l, l is this and it will give us a prime which is basically this many bits. So, now, there is no such algorithm exists which can give us a prime number of our desirable size. So, then what is the option? Option is we can just choose an integer - odd integer, and we can test whether this is prime or not. So, we can choose an integer, so that is those are called primarily tests basically.

(Refer Slide Time: 01:52)

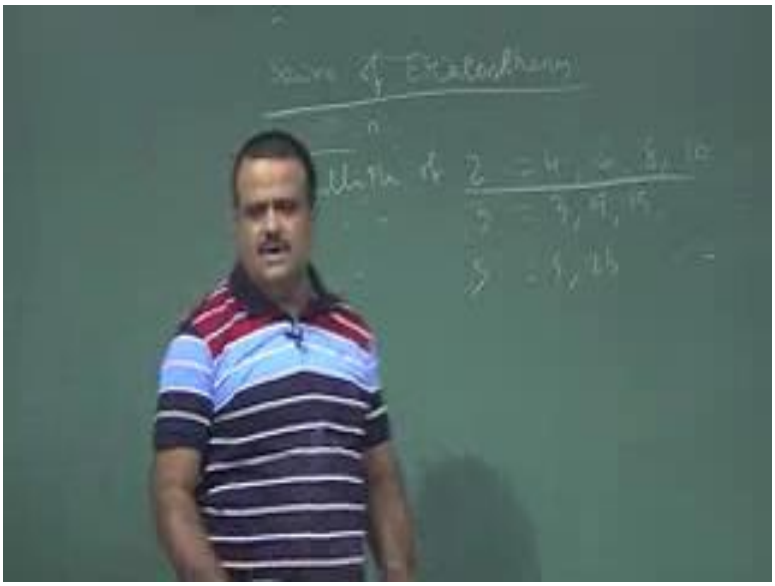


We can choose an integer p and then we check or test this checking means testing check whether this is odd integer, because 2 is usually is taking as a prime which is a even number, but any other even number cannot be prime because it has a factor 2. So, prime means definition of prime number is it is only divisible by 1 and p; so other than 1 and p it should not have any factor. So, we should not get any factor other than 1 and p.

So, now we can check whether p is prime or not. So, we have seen this we can check by suppose for checking this we have to start with this is a naive approach we can start with 2, 3, 4 like this and we can check whether this divides p or not. So, we take a k from this whether p has a factor this or not we take a k from this set and then we check whether k divides p or not. Now, this we can reduce by checking up to root over of p because we have seen if a is a factor which is greater than root over of p then so a is a factor means there will be a b such that p is equal to b then b has to be less than root over of p.

So, if we have a factor of p which will get at the root over of p then we should have a factor another factor b which is less than 2. So, it is sufficient to check this up to, we can leave it up to this. So, but this is not a good idea because if p is large, so it checking too much is so this approach is not that a good idea. So, another approaches we have seen which is called a sieve of Eratosthenes so that idea is we just remove the composite then just we like this we just remove the composite then the remaining number is prime.

(Refer Slide Time: 04:55)



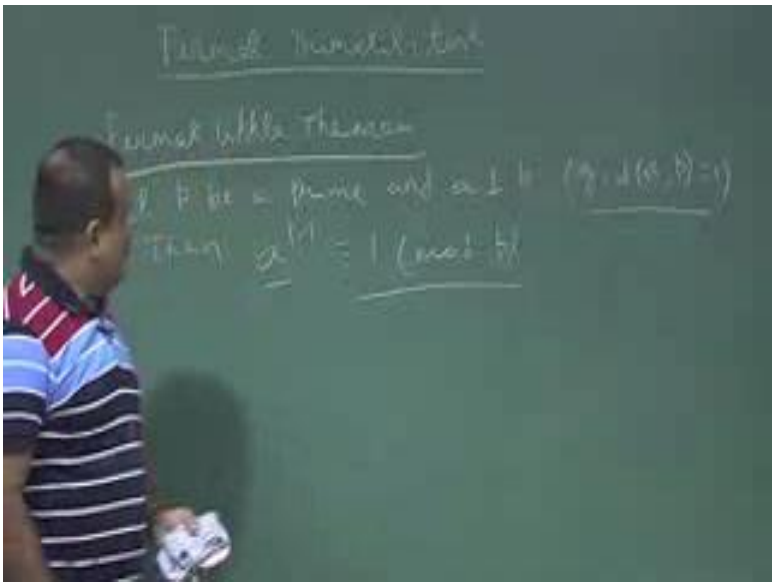
Seive of Eratosthenes, idea is we will just sieve it I mean we just delete all the composite and then the remaining are the primes. This we have seen like we first delete suppose we want to check p is prime or not; I mean we want to consider 1 to n some numbers, we want to get all the primes from 1 to n. So, what we do we delete all the, remove all the composite from 1 to n and then the remaining will be the prime number, so that is the idea. So, for deleting what we do, first we need all the numbers which are multiple up to we remove all the number multiple of two then we remove all the number which are multiple of, so multiple of 2 means 4 we delete, 6 we delete, 8 we delete, 10 we delete like this.

Then we delete all the numbers which are basically multiple of 3, but 2, 6 we have deleted 3, 9, 15 like this. So, then we delete all the numbers multiple of 4, but that will come under multiple of 2. So, then we delete all the number multiple of 5, so 5, 15 is come under 3 multiple like this, so this way. So, after this the remaining number will be the prime number. So, this is also not a very efficient approach, because the n is large. So, this way it will take much more time.

So, now, we want to see whether we can have some better approach to the better way to test whether a odd integer is a prime or not. So, for that we need to take help of some properties of prime number and then we can, so those are the necessary condition of p to be prime. And then we can use those property to test whether our selected number our

chosen number is satisfying that property. If it is not satisfying then we can straight away say this number is not a prime it is a composite number, but if it is satisfied then we can say it could be a prime. So, those are basically probabilistic of I mean those are called pseudo prime will come to that.

(Refer Slide Time: 07:42)

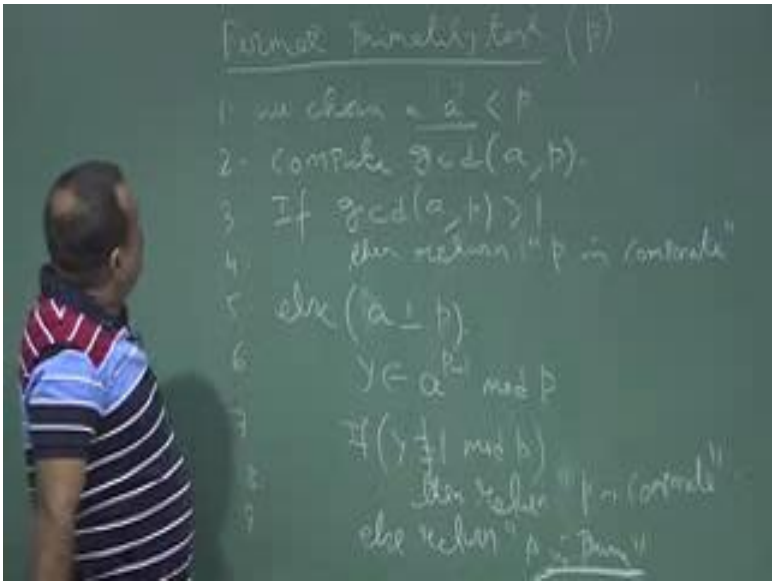


So, let us have this Fermat's primality test. So, this is coming from the Fermat's little theorem let us recall the Fermat's little theorem. So, what is the Fermat's little theorem? So it is telling let p be a prime number and a is relatively prime with g, d the g, c, d of a p is 1. Then a to the power p minus 1 is called 1 mod p and this is true for all p or all a, this must be true for all a where gcd of a p must be 1. So, this is the Fermat's little theorem. So, we want to use this. So, this is a necessary condition to p to the prime.

So, if we choose number p odd integer now, and if we take a which is basically g c we can calculate the gcd of a and p, now if gcd is not 1 then we can say p is not a prime because they have a common factor greater than 1. So, then we can say that there p is not prime, but if the gcd is 1 and if this condition is satisfying, if this condition is not satisfying then we can say p is not a prime, because this is a necessary condition to be p to be prime. But if this condition is satisfying then we say p may be a prime, so that is why it is called pseudo prime probable prime. So, p is prime base a. So, this base a is coming into the picture like because we are not testing for all a, if we can test for all a

then we can sure that p is a prime, but that test is equivalent to that the seive method we have seen. So, we will use this. So, let us just write the primality testing of Fermat's.

(Refer Slide Time: 10:12)

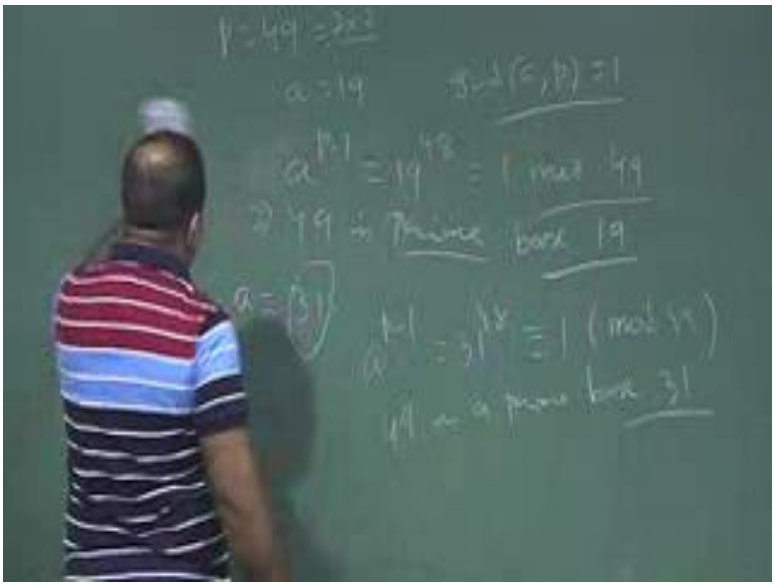


So, we choose p and we check whether p is a prime or not. So, we select a choose a such that which choose a randomly and says that gcd of a and p we compute. We choose a such that gcd of if we choose an a, and if we found gcd is not 1, greater than 1, then we can straight away say this is not a prime. We chose a less than p less than p an integer less than p then we compute gcd of a p. Now, if the gcd is greater than 1, then they have a they have a factor more than 1, so that means, p cannot be a prime then return p is composite number, not prime is composite. Else if the gcd is 1 else means the gcd of else means a is co prime to p that means, they are gcd is 1. Then what we do we compute this a to the power we compute this we store it into y a to the power p minus 1 mod p.

So, how we can compute this, this is this is just a powering a number. So, this powering we can compute we have seen for RSA, so square and multiply method we can use. So, we can one can do this powering, so we check this. And then we see whether else if y is congruent to 1 mod p then if y is not congruent to 1 mod p then return p is composite for sure. Else, if it is 1, then it is satisfying the necessary condition then p is we can return as a prime, but this is also probabilistic sense because we have only chosen one a, we are not testing all possible a. If we can test for all possible a, and if this for all a's which are having gcd with p is 1 and then if it is giving us that a to the power p minus 1 is 1 then

return this composite else return p is prime. Actually this is probable prime that means p is prime base a we can say; else return p is prime, but this prime is also probable prime. So, this is the code for a Fermat's primality testing. So, this will give us probable prime we will take some example for this.

(Refer Slide Time: 14:21)



Suppose, we want to test whether say p is equal to 49 is prime or not. So, p is equal to 49, we want to test by Fermat's primality testing whether this is prime or not. So, how to test this? So, we have to choose a for which the gcd is 1. So, if we choose a is equal to say 19, then we can compute the gcd of a and p which is basically 1, this for computing this we can use the extended Euclidean algorithm to compute this. So, we compute this. Now, if we compute this, so gcd is 1, so now, if we compute nineteen to the power p minus one p minus one is, so we compute this a to the power p minus 1. So, this is basically 19 to the power 48. So, this will be happen to be congruent to 1 mod p 1 mod 49, we can verify this, this we can calculate and we can verify this.

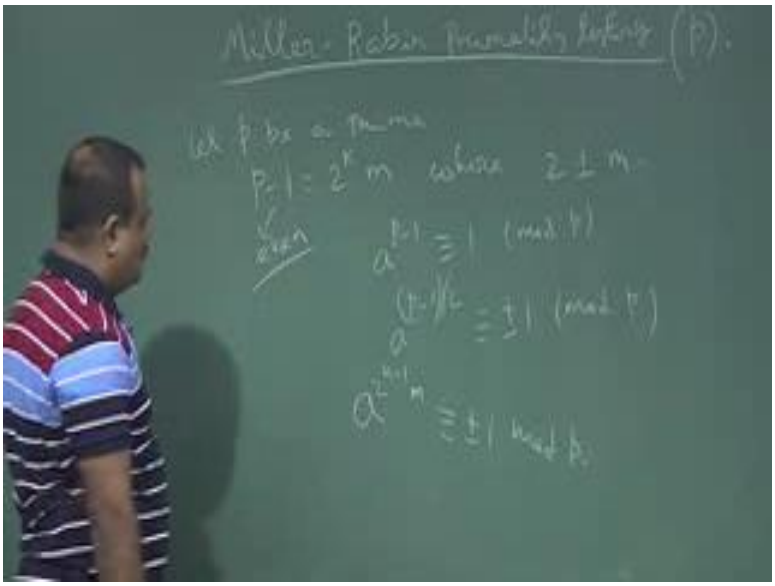
So, if we just use the Fermat's primality test, it is telling us 49 is a prime, but it should tell us probable prime, because we know 49 is not a prime, because 49 is basically 7 into 7, 49 is not a prime, but it is reporting as 49 is a prime. So, we should use here base 19, so that means, if we take a is 19 then it is reporting as a prime which should be a probable prime, but which is not a prime. So, this is a false alarm kind of thing.

Then again if we choose, so we can try for another a. So, if we say choose a is equal to say how much a is equal to say a 31. If you choose a is equal to 31 and then if we compute this a to the power p minus 1 is basically 31to the power 48. So, this also you can verify this is 1 mod congruent to 1 mod 49. So, this is also telling us for this charge of a this is also telling us 49 is a prime this 31.

(Refer Slide Time: 17:05)

So, now suppose we choose a is equal to say 5, suppose we choose a is equal to 5 then we compute a to the power p minus 1 which is basically 5 to the power 48. So, this is not equal to 1 mod p. This you can check. Then this will Fermat test will report us 49 is not a prime. So, this decision is correct, there is no probability in this design, but earlier decision probable prime, pseudo prime, prime based these a, so that is the k. So, if our primary test is reporting that it is composite. So, that is certain deterministic way we can tell this is composite number, but if our probabilistic algorithm test is telling this is prime then it could be a it should be probable prime basically. So, this is the Fermat's test primality test.

(Refer Slide Time: 18:27)

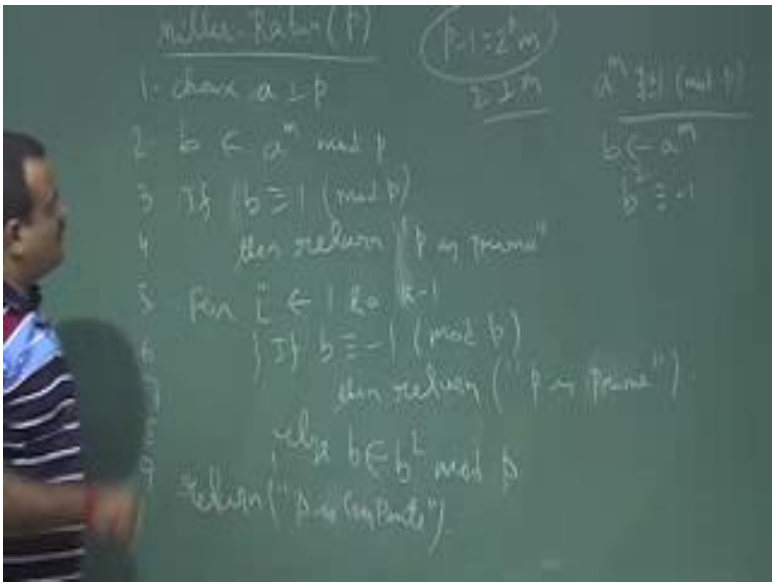


Now, we talk about Miller-Rabin primality test that is also probabilistic test. So, this is Miller-Robin. So, we have a p and we have a odd integer p and we have to check whether p is prime or not. So, idea of this test is p is odd integer, so if p is prime, so let p be a prime then what we have then p minus 1 must be odd and that should be written as 2 to the power k into m, where 2 and m are relatively prime. Otherwise, if they have a factor two again we can take that two here. So, there should be some k such that this k could be 1 also, but at least there should be one k because this is the even number because p is odd number. So, this is the case now by Fermat theorem we know k to the power p minus 1 is congruent to 1 mod p so that means, a to the power p minus 1 by 2 these are the square root plus minus 1. So, now, what is a to the power p minus 1 by 2? a to the power p minus 1 by two is basically a to the power 2 to the power k minus 1 into m is congruent to plus minus 1 mod p. So, either it is plus 1 or it is minus 1.

(Refer Slide Time: 20:32)

So, either, this can be written as a to the power m to the power k minus 1 is congruent to either it is plus 1 or minus 1. So, this is the properties of prime. So, this is the necessary condition. So, if p have happened to be prime, it should satisfy this. So, if it is not satisfying this then we can say p is not a prime number, so that is the test basically. So, we check whether a to the power m is congruent to 1 mod p or not; if it is not then we check. So, if we take this as b a to the power m then we check whether it is actually plus minus 1, then we check square of it, we check the b square is congruent to minus 1 mod p or not like this. So, this condition must satisfy, so that will give us the well that will give us a test necessary test.

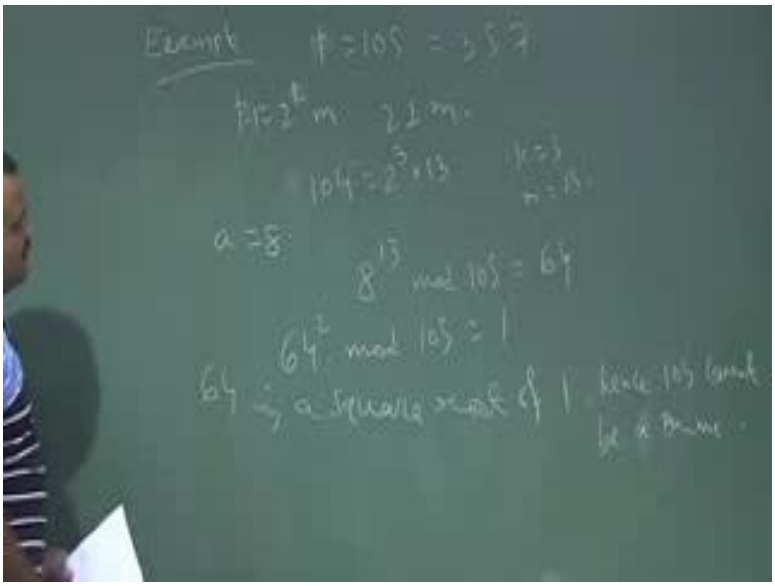
(Refer Slide Time: 21:45)



So, let us write the test Miller-Rabin primality test. So, this is this is the code for Miller-Rabin. So, p we write it as this p 2 to the power k into m, where 2 and m are gcd is 1. So, we compute b, b is a to the power m. So, again we have to choose a again we have to choose a such that gcd of a and p is 1. So, this has to be there. So, we choose a if we randomly choose a and it happened that gcd is not 1 greater than 1, then we put p is not prime. So, that part I am no writing choose a such that gcd is 1 then we compute this b, a to the power m mod p where m is this.

Now, if b is 1 mod p then we return p could be a prime, then return p that is prime, it should be probably prime or prime base a something like that we have seen. Else for because now we have to check whether b square, b q they are giving us this minus 1 or not if they are giving us minus 1 then this condition is satisfied. So, for i is equal to 1 to k minus 1 we keep on compute b. So, under this loop if b is congruent to minus 1 mod p this fast is checking whether this is plus minus 1 or not, then also we return t is composite, sorry p is prime it should be probable prime. Else so under this for loop, else we compute this b square, b is replaced by b square mod p, and then we run this loop and once we complete this loop. So, 6, 7, 8 once if it completes this loop then we return n is composite p is composite because we have tried for all plus because this is up to k, k minus 1.

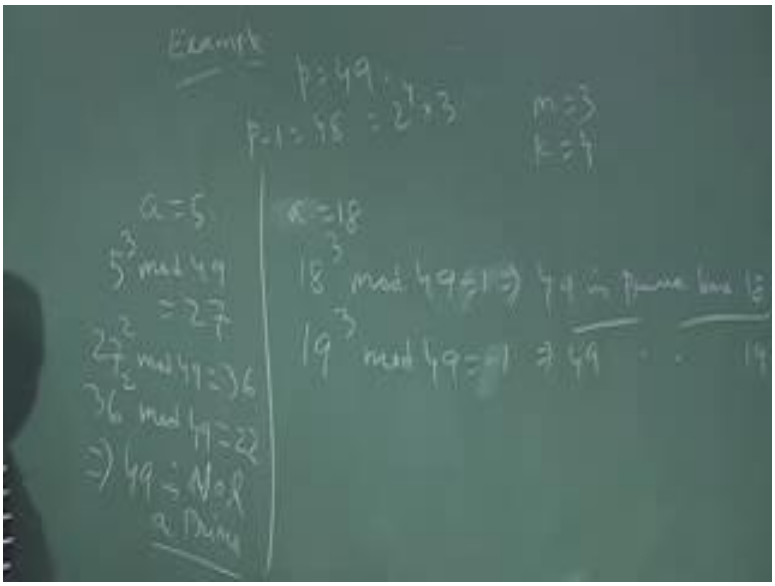
(Refer Slide Time: 25:31)



So, let us take an example quick example for testing this. If it is returning composite then it is surely a composite number, there is no probabilistic sense, but if it is returning a prime then this is a probable prime or pseudo prime base a, we can say, but anyway this is the test. So, let us take some example. Suppose we choose n is equal to 105. So, we know 105 can be written as this factor, but we are pretending that we do know this; we are unable to factorize this that we are pretending. So, now, we want to write this is sorry p. Now, we were write p to the 2 to the power k into m when m and 2 is relatively prime. So, 105 is basically, so sorry p minus 1, so p minus 1 is basically 104, so this should be written as 2 cube into 13. So, here k is 3, m is 13.

Now, we choose a say a is 8 such that there is the gcd of a and p is 1 and then we compute a to the power 13 is a to the power n mod 105. So, this is give us 64 and then we compute again 64 square mod. So, this is b. So, b square again mod 105, this is giving us 1 so that means, 64 is the square root of 1. And hence 105 cannot be a prime, so it is a composite number. Since, 64 is the square root of 1, hence 105 cannot be a prime.

(Refer Slide Time: 27:43)



So, let us take another example say this example we have seen n is equal to say 49 just now we use this example for Fermat. So, let us apply here also. Suppose we take p is equal to 49. So, what is p minus 1, p minus 1 is 48, which is basically 2 to the power 4 into 3. So, m is 3 and k is 4. So, what we do? So, if we choose a to b, but we have seen if we choose a is 18, then 18 to the power m mod 49 is 1, so which is giving us this 49 is a prime, but base 18. Similarly, if we choose a is equal to 19. So, 19 to the power 3 mod 49, so this is basically congruent to this is basically 1. So, this is also basically minus 1. So, this also is giving 49 is a prime base 19 like this.

Now, if we choose a is equal to 5 say then what happened then we compute a to the power m mod 49. So, this is our b. So, this is basically 27, then we compute 27 square mod 49, this is basically 36. Then again we compute 36 square mod that for loop, this is basically 22, now we stop because k is equal to 4, three times we have to do which is not minus 1. So, this implies 49 is not a prime this will not a prime, so that is guaranteed that is for sure 49 is not a prime. Otherwise, if it is reporting a prime, so this should be probable prime; otherwise, you have to check for all possible a.

Thank you.