

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

Lecture - 26
More on RSA

We talk about more on RSA like; so you have seen the RSA cryptosystem.

(Refer Slide Time: 00:27)



So, where Alice and Bob is communicating and Bob is doing key generation or the set up because Bob has to get the Bobs public key; and because Alice wants to send a message to Bob so Bob has to Alice need to get Bobs public key. So that Bob can only do this setup in order to get the Bob public key private keep here. For this setup we need to have say Bob has to choose two prime number p and q , such that two prime number p and q then Bob. These two primes so Bob has to choose then Bob compute p into q , and then Bob computes ϕn which is Euler's phi function and then Bob choose a e which is basically the city of which is basically co prime ϕn in order to have the inverse.

That means you will have the inverse that is called d . So, $e d$ is congruent to 1 mod ϕn it is inverse exists because we have this condition they are relatively prime. So, this is the key generation phase. So, Bob has to do this key generation and then Bob make d 2 the private sorry, d and the public key is basically d and $p q$, $p q$ also be private and public

key basically n and e and this will be sent to Alice or anybody else who wants to send a message to Bob.

So, suppose Alice wants to send a message to Bob so Alice will receive Bob public key. So now Alice will compute, so n to the power e so this is the ciphertext m to the power e mod n . And also then this is the encryption and for decryption or Bob will do Bob will compute y to the power d mod n . So, this will give us n this we have correctness we have seen. So, this is the RSA cryptosystem.

(Refer Slide Time: 03:12)



Computational Aspects

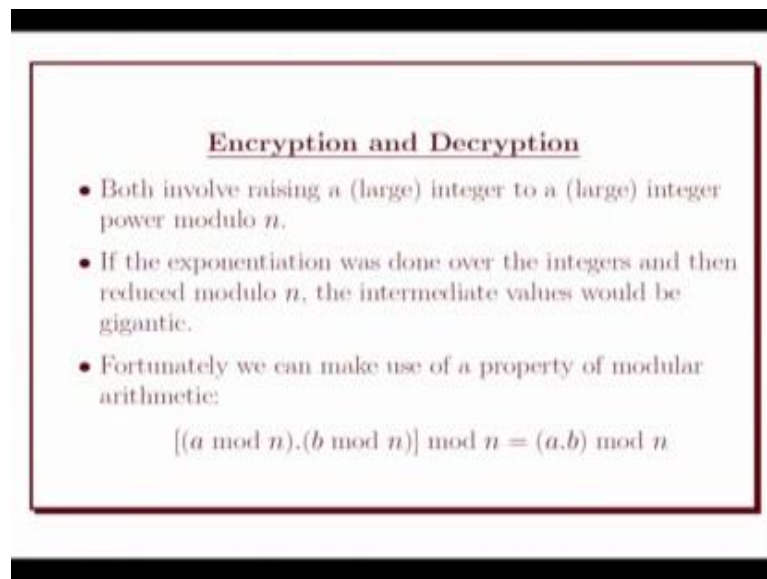
- The complexity of the computation required boils down to two aspects:
 1. The actual encryption/decryption process.
 2. The key generation process.

So, we today in this talk we will talk about some computational aspect on RSA and the security aspect on RSA. Let us start with the computational aspect of RSA. So, the complexity of the computation requires two steps like: for actual encryption and decryption. So, we have seen for encryption we need to do the n to the power e so that exponentials and how we can calculate. And for decryption also Bob has to calculate y to the power d mod n ; for encryption $x = m$ to the power e and for decryption Bob has to calculate.

So, for encryption m to the power e ; so how one can calculate this? So, m may be some large number say 191. So, e may be some another number say 9, so mod something some large number. So, how to calculate this type of exponentiation? This is in encryption as well as for decryption also, but Bob is doing good, because this is a number so this is 51123 say for example, answer d is some number say 411697 mod something.

So, this is one aspect of computation complexity. How one can calculate such exponentiation? So, that you have to handle. And another issue is we meet the primes so p, q . So, how to get the primes p, q ? So, is there any algorithm who can give us the prime p, q , so the prime numbers. So, that is the key generation step. So, this two are basically computational aspect of RSA.

(Refer Slide Time: 04:45)



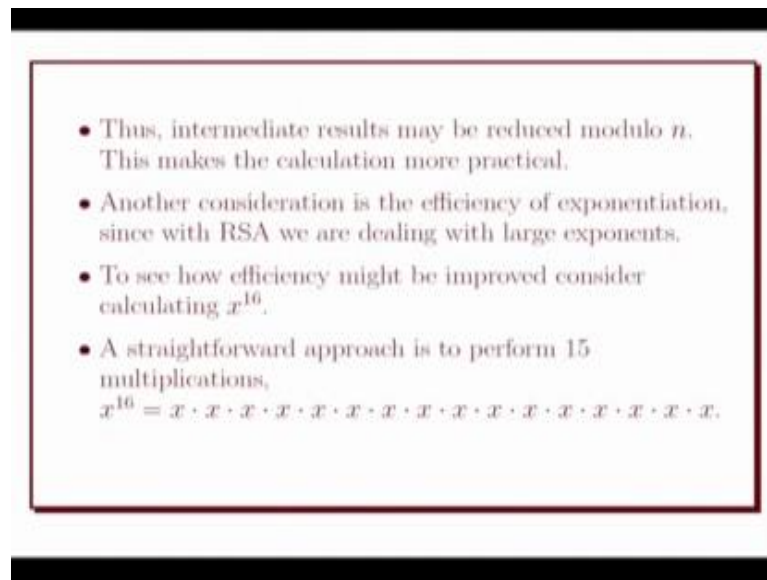
Encryption and Decryption

- Both involve raising a (large) integer to a (large) integer power modulo n .
- If the exponentiation was done over the integers and then reduced modulo n , the intermediate values would be gigantic.
- Fortunately we can make use of a property of modular arithmetic:

$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

Now, for encryption, so exponentiation we need to do. Basically you need to compute some a to the power powers or x to the power some power something like that. So, both involve some large. So, these are under modular operations. So, under modular operation we can use this fact that $a \bmod n$ into $b \bmod n$ then mod n is equal to a into $b \bmod n$.

(Refer Slide Time: 05:16)



So, suppose for RSA we need to do the exponentiation. So, suppose we want to calculate the x to the power 16.

(Refer Slide Time: 05:35)



Suppose you want to calculate x to the power 16. So, x to the power 16 is basically x into x into x 16 times or say 16 times. So, this if we need to calculate x to the power e or to say x to the power m or say x to the power n . So, this is (Refer Time: 05:55) approach. So, this will take order of n times I mean like n multiplication, if it is x to the power n

like this. So, how we can have efficient computation for this exponentiation, so that is one challenge in RSA.

(Refer Slide Time: 06:20)

- However we can receive the same result with just four multiplications if we repeatedly take the square of each partial result successively forming x^2, x^4, x^8 and x^{16} .
- Note that even utilising shortcuts etc. there is a requirement for arithmetic operations with arbitrarily large integers and most computers are restricted in this capability.

For that finding x to the power 16 what we can do we can just calculate x square then again x square into x square, so x to the power 8 then x 3 or 4 then x to the power 8 like this you can do.

(Refer Slide Time: 06:36)

- More generally, suppose we wish to find the value a^m , with a, m positive integers. If we express m as a binary number $b_k b_{k-1} \dots b_0$, then we have the following:

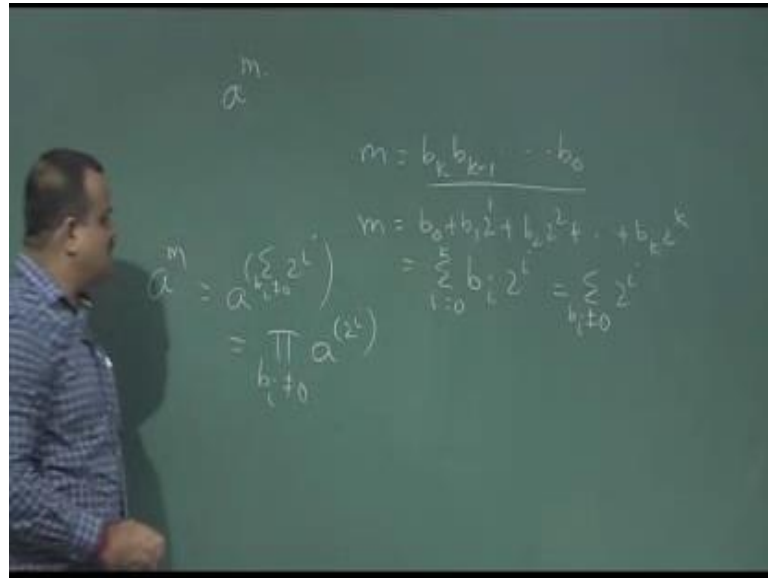
$$m = \sum_{i=0}^k b_i 2^i = \sum_{b_i \neq 0} 2^i$$
- Therefore,

$$a^m = a^{(\sum_{b_i \neq 0} 2^i)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^m \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)} \right] \bmod n = \left(\prod_{b_i \neq 0} [a^{(2^i)} \bmod n] \right) \bmod n$$
- Which can be done using a square an multiply algorithm.

So, this is called, this technique we will use in what is called square and multiply method. So, more general suppose you want to find a to the power n , so a b are two positive integer. Now we express m in a binary form, like this.

(Refer Slide Time: 06:56)



Suppose we want to calculate a to the power m . So, our a and m are two positive integers now if express m is in binary say $b_k b_{k-1} \dots b_0$, k bit binary bit. Now, m will be written as what? M will be written as b_0 plus $b_1 2$ plus $b_2 2^2$ square like this; that means, $b_k 2^k$ to the power k . This is in decimal integer the m integer. So, to get the binary form we have to divide keep on divided by 2. So, you got b_0 then we again divided by 2 on the remainder like this we want to do like this. So, this is basically you can denote by some of the $b_i 2^i$ to the power i ; i is equal to 0 to k .

Now we are calculating a to the power m . So, a to the power m is basically; this we can just write a to the power, so summation of $b_i 2^i$. So, this we can write just summation of 2^i to the power i where b_i is not equal to 0; so 2^i to the power i where b_i is not equal to 0 that is it.

This sum will take the product form, so a to the power two to the power i . So, this b_i is not equal to 0, where b_i is not equal to 0. So, every time we will square it and once the b_i is not 0 will multiply it, this is called square and multiply. So, this will give us this expression like this. So, if m is this then a to the power m is basically in this form then this is basically product from a to the power y . So, every time you square it, so a to the

power a 2 i, so this is basically model is basically in this form; so this is called square and multiply method, square and multiply algorithm.

(Refer Slide Time: 09:25)

- The square-and-multiply algorithm to compute $a^m \bmod n$:
 - $z = 1$
 - for** $i = k$ **downto** 0 **do**
 - $z = z^2 \bmod n$
 - if** $b_i = 1$ **then** $z = z \times a \bmod n$

So, this will write in a pseudo code like this, so we start with z is equal to 1, z is that our a to the power m. So, now we start with that b k. So, this is b k down to 0. Now every time we square the z. So, z is equal to z square model and if the b i is 1 then only we multiply with a. So, b i is 1 then only z is equal to z into a mod n. So, every time you are squaring only will multiplies when v i is equal to 1.

(Refer Slide Time: 10:01)

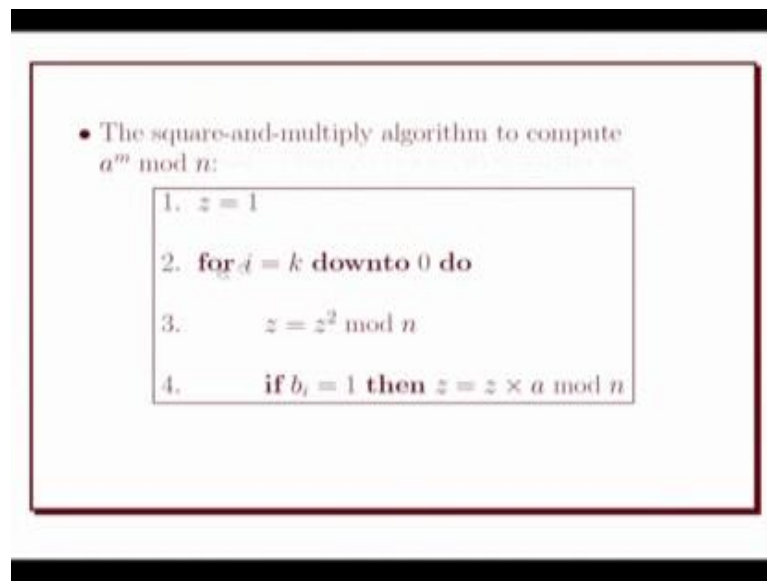
- Let $n = 11413$, $m = 3533$ and $a = 9726$. We now compute $9726^{3533} \bmod 11413$ using square-and-multiply algorithm.

i	b_i	z
11	1	$9726^1 \bmod 11413 = 9726$
10	1	$9726^2 \bmod 11413 = 2659$
9	0	$2659^2 \bmod 11413 = 5038$
8	1	$5038^2 \bmod 11413 = 6167$
7	1	$6167^2 \bmod 11413 = 4568$
6	1	$4568^2 \bmod 11413 = 5783$
5	0	$5783^2 \bmod 11413 = 6386$
4	0	$6386^2 \bmod 11413 = 4629$
3	1	$4629^2 \bmod 11413 = 10340$
2	1	$10340^2 \bmod 11413 = 105$
1	0	$105^2 \bmod 11413 = 11035$
0	1	$11035^2 \bmod 11413 = 5761$

- $9726^{3533} \bmod 11413 = 5761$

So, here is an example. So, if you take n is equal to say some value 11413 this is in and if you take m is equal to 3533 and a is equal to 9726. And suppose you want to compute $a^m \bmod n$ using the square and multiply method you have seen. So, we write this m into binary form, so this is the binary representation from b_0 to b_{11} . And then we call this algorithm we start from b_{11} and every time we square we start with z is equal to 1, every time square whenever we see a 1 in b_i we multiply.

(Refer Slide Time: 10:42)



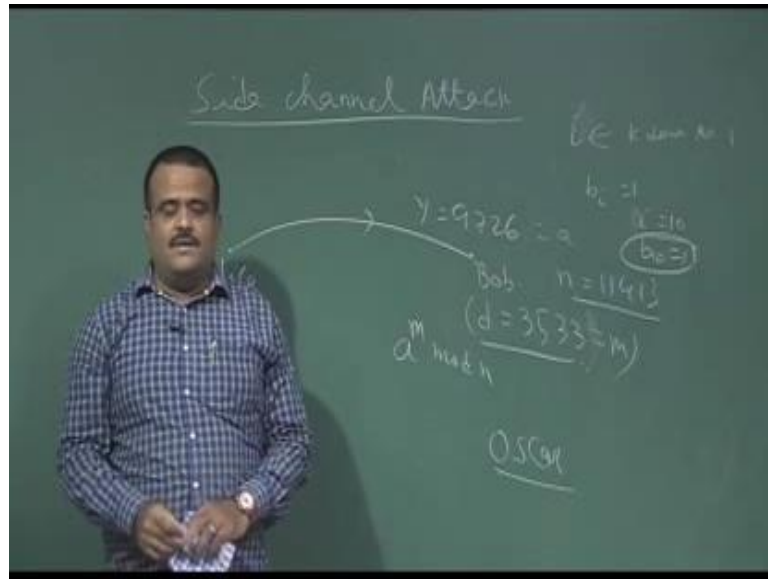
• The square-and-multiply algorithm to compute $a^m \bmod n$:

1. $z = 1$
2. **for** $i = k$ **downto** 0 **do**
3. $z = z^2 \bmod n$
4. **if** $b_i = 1$ **then** $z = z \times a \bmod n$

So, this z is basically 1. Now this is 1, we have to square the z , so 1 square and this is because this is 1 we have to multiply. So, this is this then again b_i is 1. So, again we square this z now z is this one. Now again we square z and then we multiply with it we got this and every time it has to be mod m . So, every operation is under mod m . So, this into this mod n will give us this. So, now, b_i is 0 we do not need to multiply with a square it z . So, if you square this z it will be d square then mod this will give us this. So, this way we continue finally we got this 5761 as the result that z . So, this 5761 is basically m to the power 8 9726 to the power 3533 mod 11413.

So, these are called square and multiply methods. So, every time we square z and once we have a once you see a 1 we just multiply. So, this method has a problem in terms of what is called side channel attack.

(Refer Slide Time: 12:19)



So, it is says side channel attack. So, we will talk about the attack in more general we will discuss the crypt analysis. For the time being suppose this is Alice and this is Bob; and suppose Bob received this y to be this number a say 9726. And say Bob secret is say d is equal to say this m 3533. This is the secret of Bob, this is our n in our square, and multiply method and this is our m in our square, and multiply method. And so now Bob need to calculate this is the m which is the secret key of Bob. Now Bob need to calculate a to the power m mod n and suppose n is in we are taking the example that we have seen 11413.

So, suppose in Bob's machine Bob is calculating this using the method just now we have described this one that square and multiply method. So, Bob is having this secret key and Bob is receiving this ciphertext, now Bob need to calculate this in order to get the message back. So, this is suppose the plaintext. So if Bob calculates this, if Bob executes this algorithm in Bob machine then suppose Oscar is sitting in the side of Bob machine. Suppose Oscar is sitting here, Oscar is sitting here and Oscar is just watching the Bob machines functionality. In the box machine this code is running, this calculation is going on. Now Oscar is sitting, Oscar knows that the Bob is using the square and multiply method. So, Oscar knows that every time it is squaring and multiplication will be done except once this bits has won.

So, Oscar will check the timing, how much time it is taking in the Bob machine or how much power consumption it is doing. When Oscar will see that some loops the power consumption this is basically a loop for loop; we have seen this algorithm has for loop. So, this is for loop it is starting from this, so this is basically for loop. In this for loop if it is square it will take similar time, but if it has to multiply. So, this operation square and multiply then the corresponding b_i must be 1.

So, that measurements Oscar can either by seeing the timing or the power consumption, because more operation means more power consumption. So, in the Bobs machine Oscar can fit a machine to measure the temperature or to measure the time. Then by seeing this how much time it is striking, suddenly Oscar see in some loops in some b_i . Say for some loop this is a loop for k down to 1 and then we are doing every time squaring and sometimes you are multiplying it b_i is 1. Suppose for 1, i is equal to 1 to 10 Oscar is saying it is taking more time; that means, corresponding b_{10} must be 1.

So, Oscar will just measure the by sitting side, the Oscar will just measure the time of the computation by either putting some machine like the power machine or something. So, this is also called timing attack. Then Oscar will give this bits are 1 basically, because this is state taking more time because it is doing one more operation multiplying operation. So, then Oscar can gives these bits. This is 1, 1, 1 like this.

So, this is called side channel attack we will talk about this on more details when we talk about the crypt analysis. So, this is my square and multiply method has a drawback.

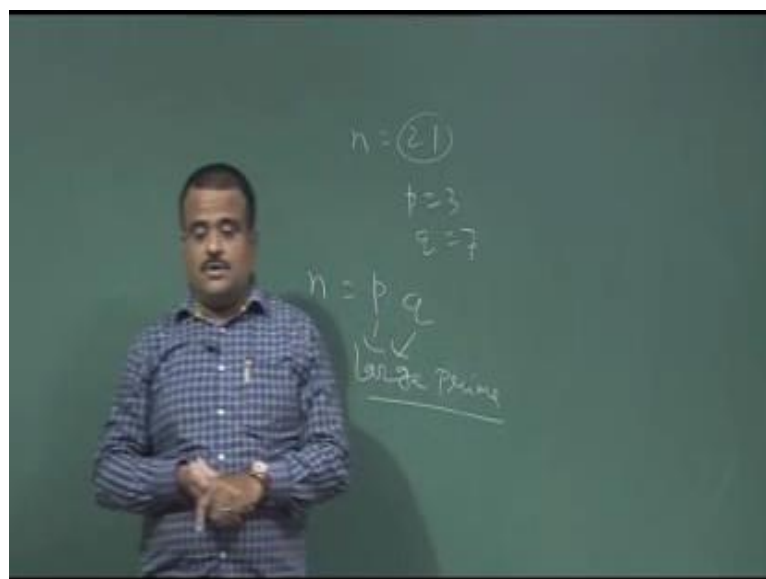
(Refer Slide Time: 16:49)

Key Generation

- Before two parties can use a public key system, each must generate a pair of keys. This involves the following tasks:
 - Determining two prime numbers p, q .
 - Selecting either e or d and calculating the other.
- Firstly, considering selection of p and q .
- Because the value $n = pq$ will be known to any opponent, to prevent the discovery of p, q through exhaustive methods, these primes must be chosen from a sufficiently large set (must be large numbers).

Now, we will talk about another aspect of computation complexity of RSA which is called key generation. So, for key generation we need to have a prime number p, q which is and we compute n is equal to p, q and we are making n to be public. This means the RSA security is based on the factorization is hard. So, if we choose say n is say 27 sorry, 21.

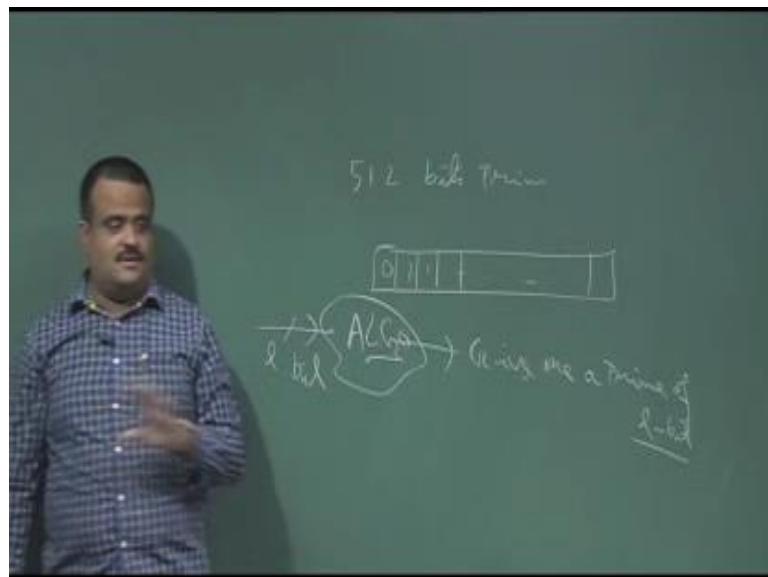
(Refer Slide Time: 17:32)



Then this one can easily guess that p is equal to 3 q is equal to 7. So, then it is not secure because once we know the p q then we know ϕn then one can guess the e ; e is known so one can guess the d .

So, that is why we cannot take n to be so low number. So, you have to choose n to be large number in order to be n to be large number then p q to be large. So, we need to have to prime number p q so these to be large prime. So, for key generation we need to have a prime number not only prime large prime say 512 bit prime. Now the question is where from we can get this time, so that is one issue of computational complexity how to get a prime.

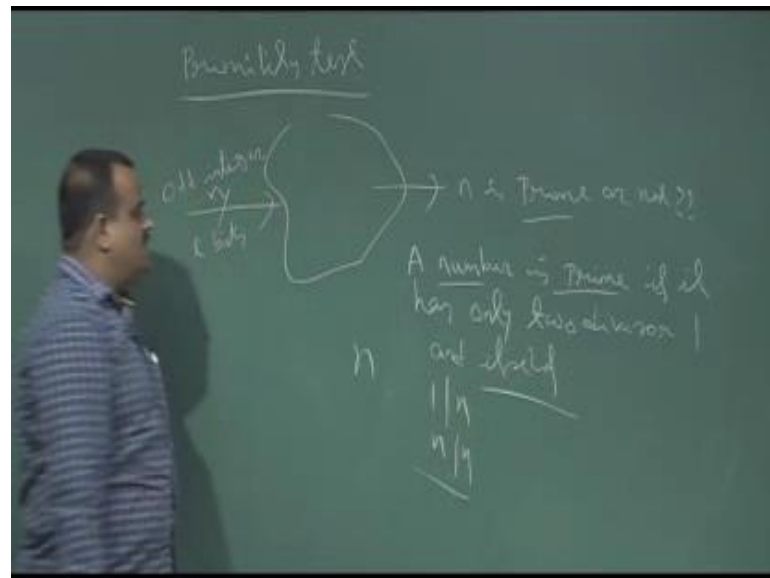
(Refer Slide Time: 18:25)



Suppose Bob needs a 512 bit prime. So, Bob need a cube p q which is 512 bit 0 on bits. So now the question is there any algorithm which can give us this prime I mean 512 bit prime or any bit. So, we will take give the input 1 bit and we can ask give me a prime; give me a prime of 1 bit.

So, is there any algo which can do that? Is there any machine? Or is there any algo which can give the input 512, give me a prime number of 512 bit? No, there is no such algorithm exists in the literature which can give us a prime number. So, what we can do we can test the, we can do the primary test. We can choose a odd integer then we can test whether this is a prime or not. So, those are called primary test.

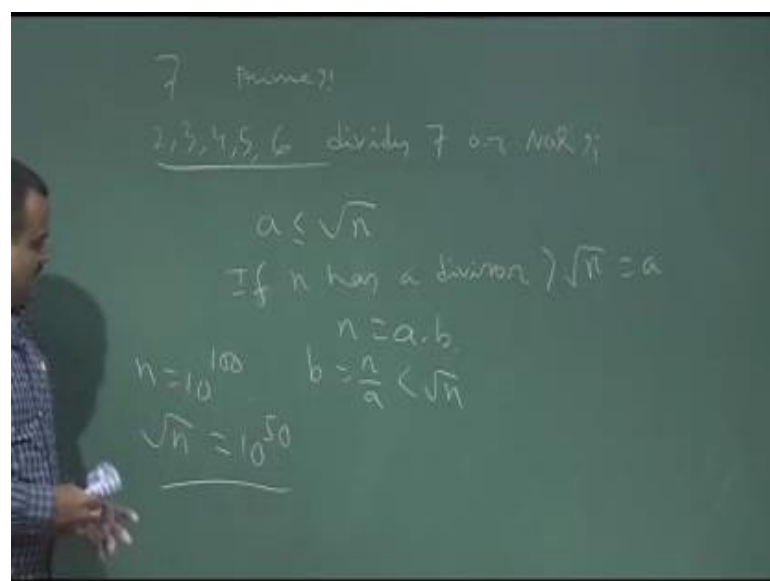
(Refer Slide Time: 19:37)



So, basically we can give an odd integer n of 1 bit say and then this algo can tell us whether n is prime or not.

Suppose I ask you is the number 7 is prime. So, when a number is prime if it has only two divisors g 1 and itself, then we call the numbers to be prime. So, if n is prime so that means n has only two divisor 1 and n is same. That means, there is no other divisor of n . So now how to test 7 is a prime?

(Refer Slide Time: 21:06)



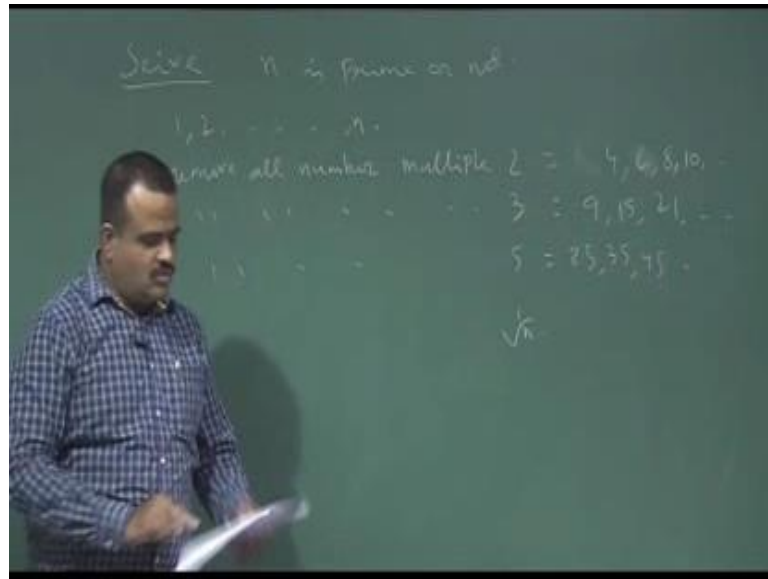
So, if you want to test 7 whether it is prime or not. So 7, is 7 is prime? So, how to case that? That means, we want to check whether the 7 has any factor or not other than 1 and 7; that means, you have to check whether 2, 3, 4, 5, 6 this divides 7 or not. See if we can check that then we are done. So, no numbers divides 7 so that means 7 is a prime.

If instead of 7 if we have some big number like n , so do you need go up to 6 or we can just go to the square root of 7. So, this method we can go up to the by checking square root of n to check n to be prime. So, we can go this method a is less than equal to up to square root of n . Why, because if n is has a factor which is greater than; if n has a divisor greater then square root of n which is say a . So n will be written as a in to b . So, a is divisor so b is another factor so then b must be b is equal n by a . So then b must be less square root of n .

That means, if n is not a prime then there has to be a divisor which is less than square root of n . So, for this checking we do not need to go for up to 6, we can go for up to square root of 7. That means, square root of 7 is 2. So, we can just check whether 2 divide 7 or not. So this is the nave approach, this is work perfectly fine. This is a deterministic approach, so this way we can tell the number is prime or not.

So, problem with this method is if n is big number, suppose n is say 10 to the power 100 then square root of n is basically 10 to the power 50. So, up to this much number you have to check for finding this is prime. So, this is not a very efficient method to check whether it is prime or not.

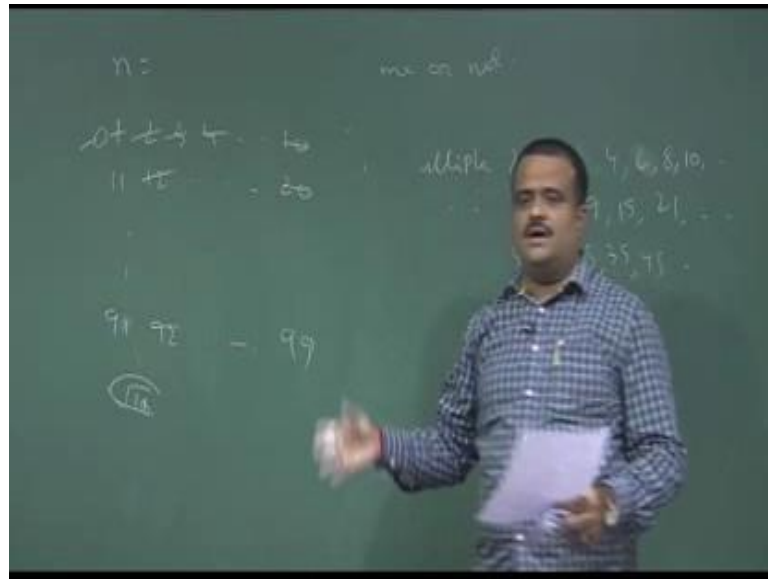
(Refer Slide Time: 23:54)



Another approach could be received this is the Sieve approach. So, the idea is something we want to test whether n is prime or not. This is sieve error for same. So, this is basically to test whether n is; the idea is we consider all the numbers from 0 to n and then we remove all the composite numbers. And then slowly the remaining number is basically the prime numbers. So, the idea is we just take all the numbers say from 1 to n and then we remove all the composite number like this way; we first remove all the number which is basically multiple of 2. Then we remove all the number which is a multiple of 3; multiplex 2 means to sorry for a 4, 6, 8 10 like this.

Then we remove all the numbers which are multiple of 3 and which is not removed by 2. So, this is basically 9, 15, 21 and so on. Then we remove all the number which are multiple of 4, but do that is covered on that 2 multiple of 2, so we remove all the number multiple of 5; 45 like this. So, these why we will continue up to routine, because we are checking. So, this way we can just removing all the numbers which are composite.

(Refer Slide Time: 25:53)



That means, if we just take n is equal to 100, so 1, 2, 3, 4, 10, 11, 12, 20 like this then 90, 91, 92 like this 99. So now first we remove or if we start from 0 and 1, so we remove 0 and 1, then we remove all the number multiple of 2 like this, then we remove all the number multiple of 3 like this. Then this way we continue up to square root of 100. Then the remaining number left out in this table is the prime number.

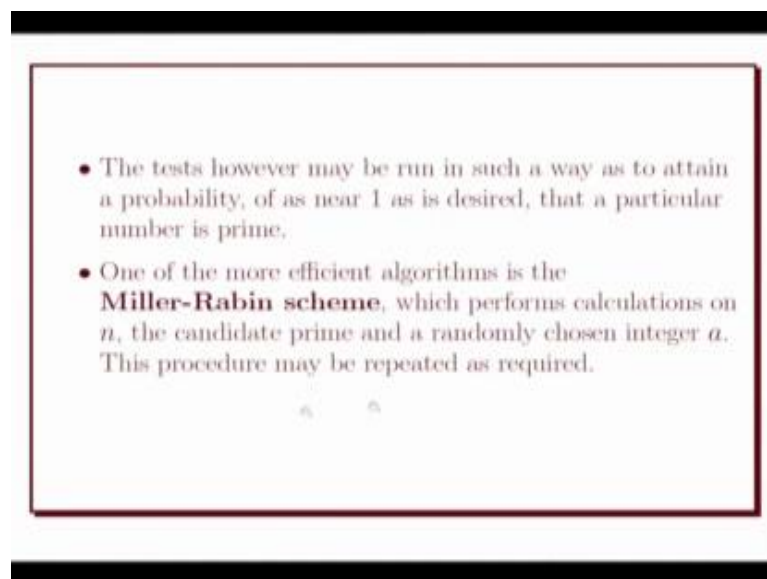
But this method is also not so efficient because if n is large, this way the removing the composite is not so efficient idea. So now, there are some probabilistic approach. So, these are all deterministic approach now there are some probabilistic methods to test the primary test; so those are published approach. So, it may give us a prime number may not. So, the success probability is there.

(Refer Slide Time: 27:02)

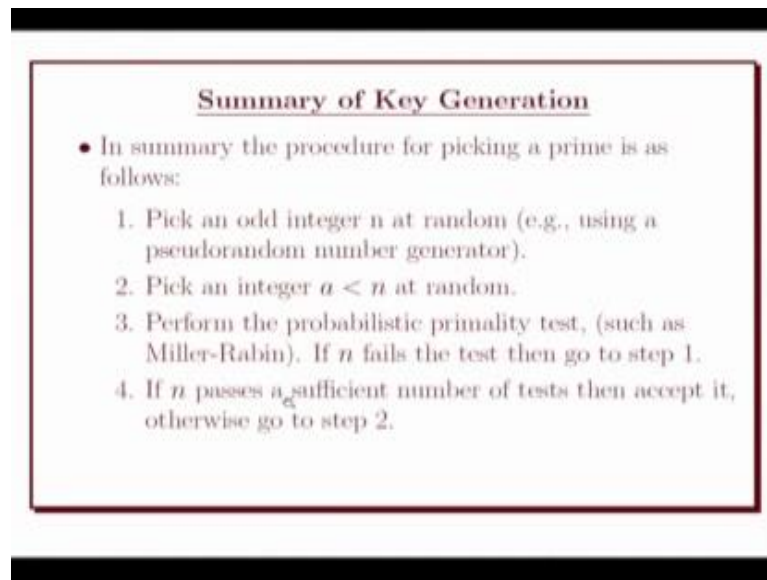


So, those tests are basically Euler's test then (Refer Time: 27:08) Solovay-Strassen test; we will see some of the taste in you know future talk. And then the Miller-Rabin test and maybe some more tests, but those are all for probabilistic test not the deterministic test.

(Refer Slide Time: 27:30)



(Refer Slide Time: 27:33)

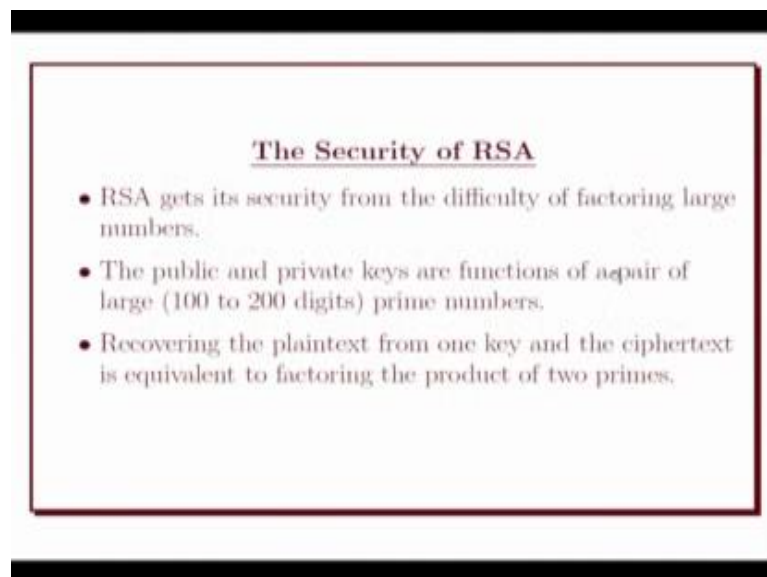


Summary of Key Generation

- In summary the procedure for picking a prime is as follows:
 1. Pick an odd integer n at random (e.g., using a pseudorandom number generator).
 2. Pick an integer $a < n$ at random.
 3. Perform the probabilistic primality test, (such as Miller-Rabin). If n fails the test then go to step 1.
 4. If n passes a sufficient number of tests then accept it, otherwise go to step 2.

So these are the test we have to run. This is basically summary of key generation for the prime. So, we pick an odd integer and then we have to perform the probabilistic primary test like we can use the Miller-Rabin test. And if we false then you have to choose another we have to try for another number. So, these are all to generate the primes.

(Refer Slide Time: 27:59)

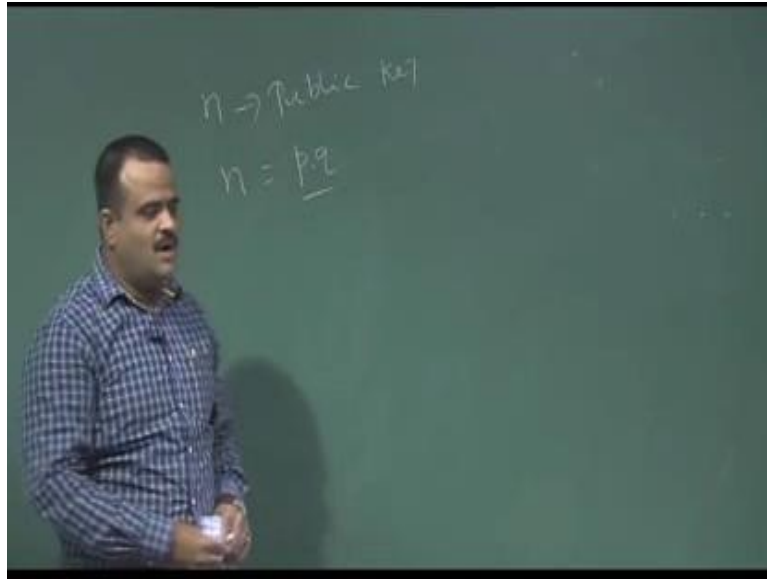


The Security of RSA

- RSA gets its security from the difficulty of factoring large numbers.
- The public and private keys are functions of a pair of large (100 to 200 digits) prime numbers.
- Recovering the plaintext from one key and the ciphertext is equivalent to factoring the product of two primes.

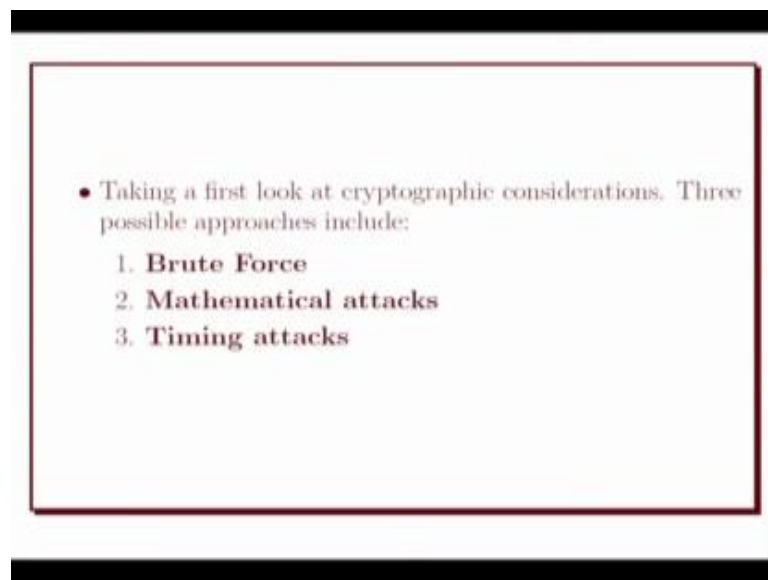
So, this is the way how we can generate the way. Now come to the RSA security of this crypto system. How RSA is secured? Why RSA secure? So, RSA is based on the factorization is hard.

(Refer Slide Time: 28:18)



That means, n is public key of, so n is some $p \cdot q$ we know, but only thing we should not be able to factor it. So, that factorization must be hard.

(Refer Slide Time: 28:37)



(Refer Slide Time: 28:50)

1. Brute Force

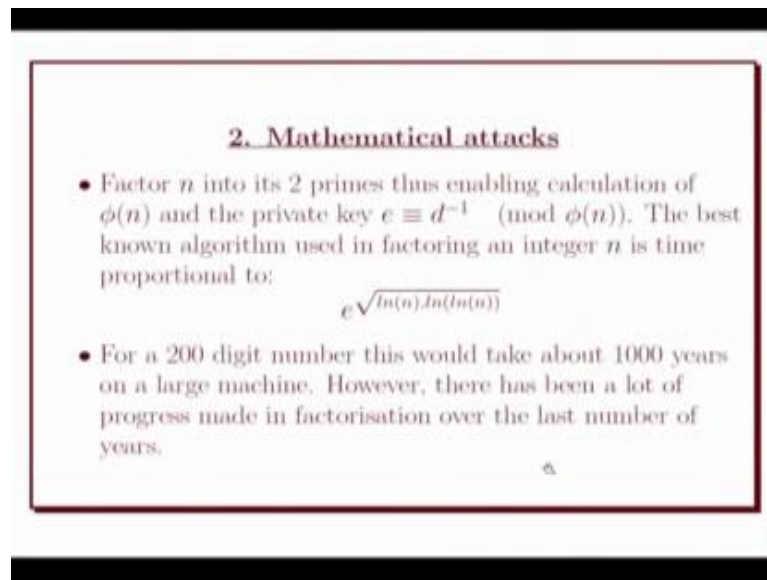
- Try all possible keys. Standard defense is a large key space.
- The larger e and d are the better, so we have the following:

	5 years ago	Today
Casual use	384 bits	768 bits
Commercial use	512 bits	1024
Military Spec.	1024 bits	4096 bits
- Where the military specification is only an estimate due to this information being classified. For comparison, 512 bits is about 150 decimal digits.

So, for this purpose we can try to factorize; so these are the attacks on RSA like, brute force attack, mathematical attack, timing attack. So, timing attack means that square and multiply has seen some side channel attack. So, for brute force attack these are the parameter RSA should have in order to be c q. So, this numbers are basically key size are this many bits for our applications like military application we need to have the key size should be this.

So, our prime number should be this many bits which is very challenging to get they this many bits prime number. So, these are under brute force methods.

(Refer Slide Time: 29:22)

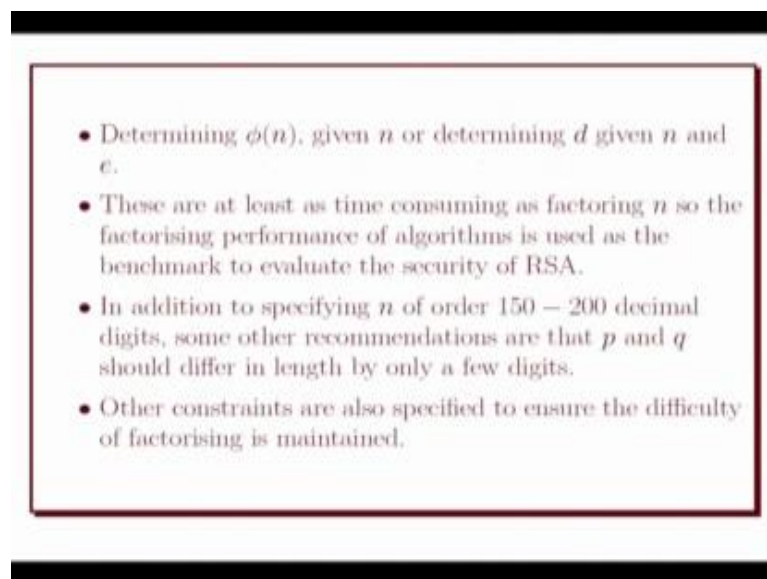


2. Mathematical attacks

- Factor n into its 2 primes thus enabling calculation of $\phi(n)$ and the private key $e \equiv d^{-1} \pmod{\phi(n)}$. The best known algorithm used in factoring an integer n is time proportional to:
$$e^{\sqrt{\ln(n) \ln(\ln(n))}}$$
- For a 200 digit number this would take about 1000 years on a large machine. However, there has been a lot of progress made in factorisation over the last number of years.

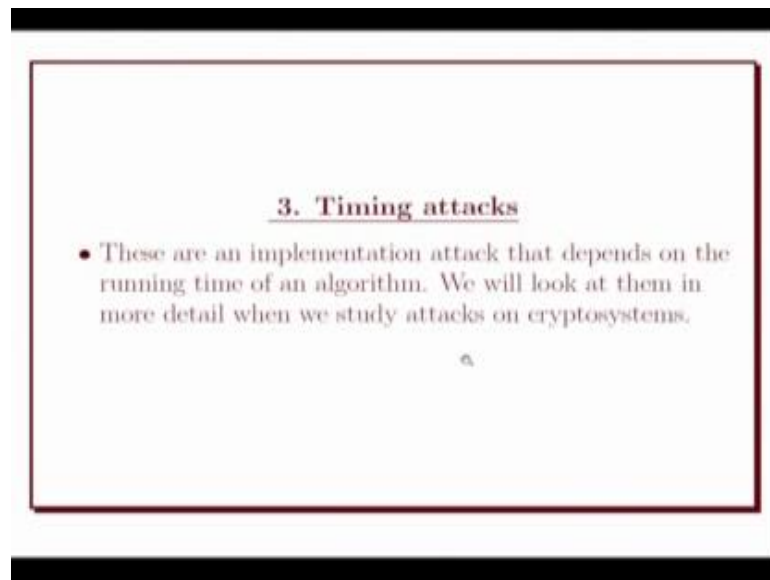
So, there are some mathematical methods. So, this is the best known method for polite is one method for doing the factorization, this is the best method for doing the factorization; this time complexity.

(Refer Slide Time: 29:42)



- Determining $\phi(n)$, given n or determining d given n and e .
- These are at least as time consuming as factoring n so the factorising performance of algorithms is used as the benchmark to evaluate the security of RSA.
- In addition to specifying n of order 150 – 200 decimal digits, some other recommendations are that p and q should differ in length by only a few digits.
- Other constraints are also specified to ensure the difficulty of factorising is maintained.

(Refer Slide Time: 29:45)



And then we have seen the timing attack. So, timing attack is basically we have seen that if Oscar is sitting in front of the Bob's machine and if Bob is doing the square and multiply method for calculating exponent then Oscar can is the key.

Thank you.