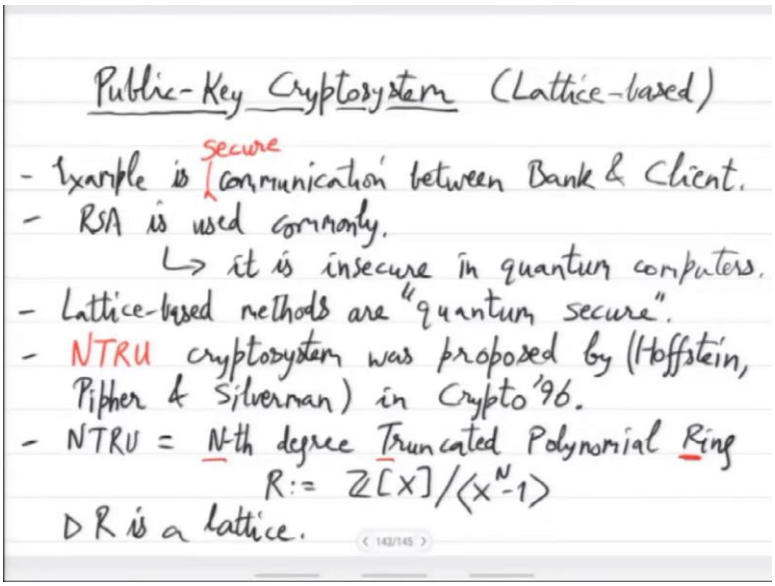
Computational Number Theory and Algebra
Prof. Nitin Saxena
Department of Aerospace Engineering
Indian Institute of Technology - Kanpur

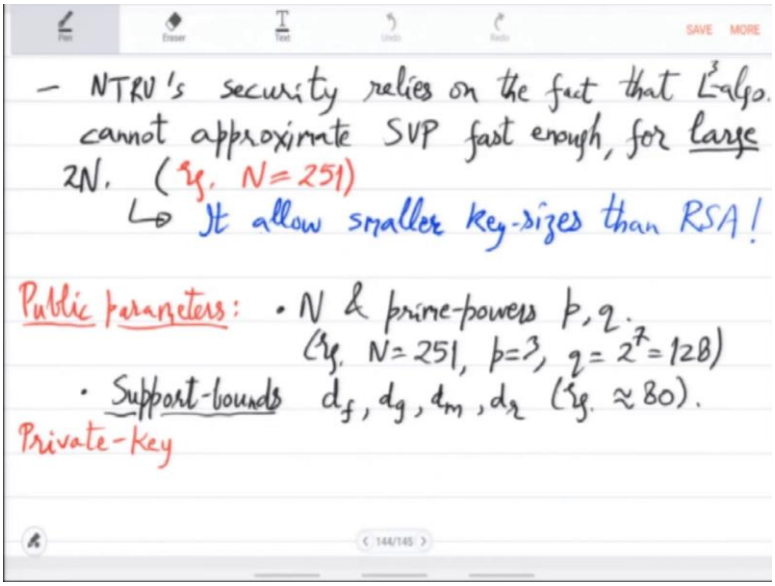
Lecture – 23
NTRU cryptosystem (continued) and Introduction to Primality Testing

(Refer Slide Time: 00:21)



So last time we started a public key crypto system that is lattice based it is called NTRU which stands for Nth Degree Truncated Polynomial Ring. So the ring R is this integer univariate polynomial ring and you mod out by Zx to the x N – 1. So this is the ring and this is actually an integral lattice. Since the underlying coefficients come from integers this is a lattice of dimension N.

(Refer Slide Time: 01:11)

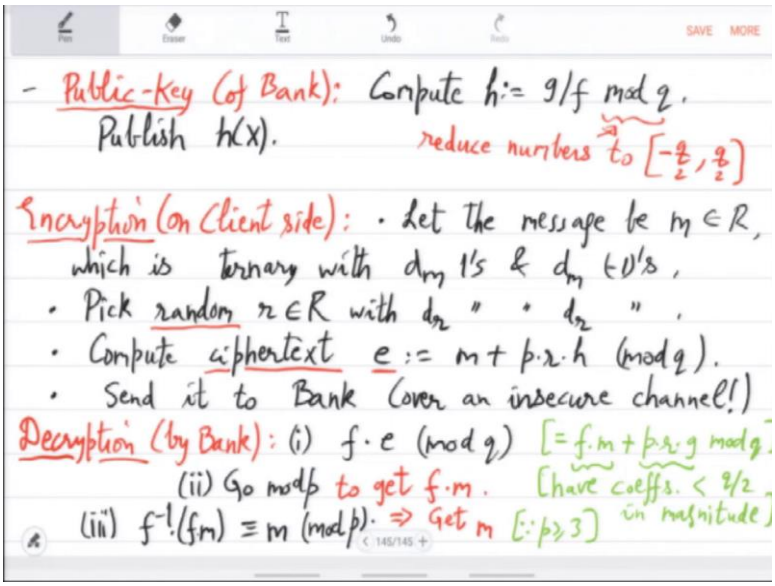


So the advantage of this over other standard public key crypto system is that the key the security really depends on the size of N the magnitude of N and it is already decently secure if you take N to be 251 or more and that relies on the fact that L cube algorithm to find reduced basis to approximate SVP will take a lot of time for N equal to 251 it will take something like 10 raised to 18 steps.

We will see that let us first describe the protocol. So we are thinking of a pair of bank and a client. So bank will publish the public parameters but will hide the private key public parameters are this number N and prime powers p and q for this discussion you can think of N to be 251 p to b 3 and q to be power of 2 like 2 raise to 7 128. Then there are some support bounds there are these four parameters which will be in practice 80 or more.

So d f, d g, d m, d r so private key is private key of the bank this is pick a random polynomial in R so 1 will be f with df many 1's and 1 less -1's and g will be d g many 1's and that many -1's so in particular note that f at 1 is 1 while g at 1 is 0. So, this is private and so f and g are polynomials where coefficients are 0, 1 or -1. So these are ternary polynomials.

(Refer Slide Time: 04:41)



And then bank actually publishes one thing more which is g by f mod q this is computed by the bank and then made public. So note that knowing h and q it is not really clear how will you find g and f it is not immediate and mod q will always mean that you get a remainder between minus plus q by 2. So let us now start with the first main step which is encryption that will be done by the client.

So encrypt suppose let the message be again an element in the polynomial ring which is m we are calling it m. And we will assume without loss of generality that which is also ternary which is ternary with the d r 1's and d r - 1's. So message also we can assume has this form this is easy to assume because you can always come up with a padding technique to make the plus 1 and the - 1's equal to this public parameter d r and the other coefficients are 0.

So that is the message that client wants to send to the bank in a secure way. So what the client will do is she will also pick a random element in the ring R sorry this is I should call it d m with an r will have d r 1's. So pick a random r so you have used all the parameters 4, 4 parameters d f, d g, d m, d r and client computes the ciphertext from m which is basically just add something some garbage to m the garbage is this and go mod q.

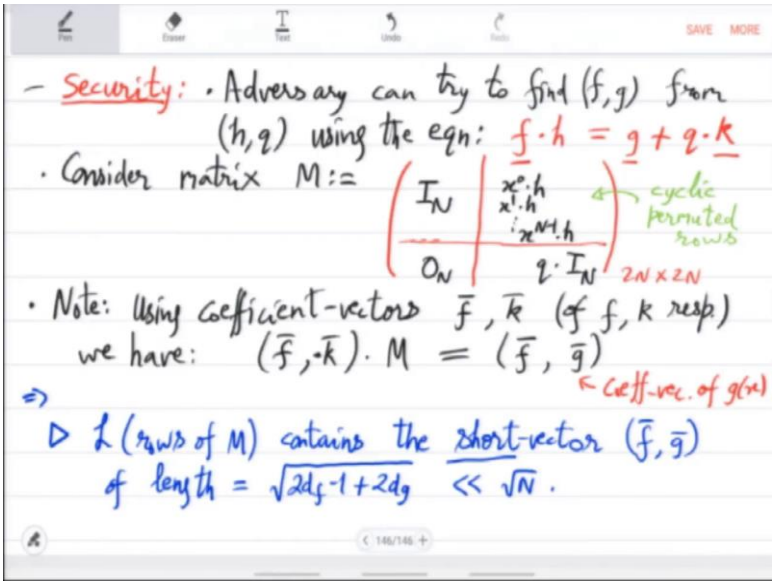
So that is the message that is the string e which will be sent to the bank this is a ciphertext. Now since an adversary will not know r it seems fair to say that for an adversary finding m from e does not look easy so send it to the bank over an insecure channel. Now how will the bank find out m from e so that is the part of this decryption?

So the bank will use the following steps. Bank will multiply e with f mod q. So what is f times e bank obviously knows f that is a private key. So this is f times e is f m + f times p times r times h f h is g so this is prg. So that is step 1. In step 2 go mod p now what happens when bank computes f times e reduce it mod q and then further reduce mod p. So, note that this both these things both of them have small coefficients.

Now since the coefficients are already smaller in magnitude than q by 2 mod q will have no effects and when you reduce mod p you will get exactly f times m. So bank has computed f times m. So once the bank has f times m bank will do what? Bank should compute f inverse times f times m mod p which will be m mod p. Now remember that m had coefficients m was ternary so I made coefficient 0 or + -1 p is at least 3.

So you will exactly get m back. Since p is at least 3. So no information is lost and bank will get m adversary hopefully will not be able to compute m from e but bank can using the key. So what can the adversary do adversary knows the public parameters in particular q and adversary knows h. Now from q and h adversary may try to find g by f right.

(Refer Slide Time: 13:22)



So in that L cube may help could L cube help the adversary. So let us check the security of this adversary can try to find f, g from h, q using the equation f times h is equal to g + q times some k. So here f is unknown g is unknown k is unknown only h and q are known. But this you can write as a or the adversary can write this as a lattice system. So consider matrix m which is it will be 2n cross 2n matrix in block form it will look like this I N and here 0 q times I N.

And here right essentially the other thing which is known to the adversary h in particular write x 0 times h x 1 times h dot dot x to the N - 1 times h look at this matrix in block form so the rows in this top right part first row is just coefficients of h and then the second row is coefficients of x times h and the last is x to the N - 1 times h coefficients. So these are just remember that you are in this ring x to the N - 1 mod x to the N - 1.

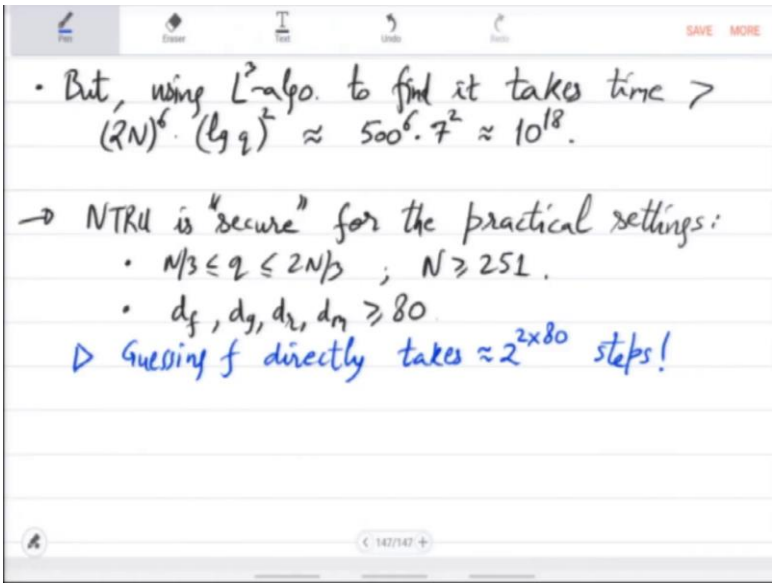
So cyclically permuted rows so that is the matrix m, so note that using coefficient vectors f bar k bar of f, k respectively we have the equation so f bar k bar times the matrix m sorry it should be - k bar. So f bar and - k bar so f bar is a row followed by the row - k bar and on the left when you multiply m what you get is you first get f bar because of the identity matrix and 0 and next you will get you will essentially get this f h from the top part and then - q k from the bottom part which is g bar so that is the coefficient vector of g.

So the point being that row combinations using integers of m gives you f and g so f and g in other words is in the lattice. The lattice generated by the rows of m contains the short vectors or the contains the short vector f bar g bar f bar g bar is short because its entries are just + - 1

and they are only d f, d g many of length so that is the number of +1, -1 and then you take a square root. So which will be much smaller than this is much smaller than square root N.

So in the ambient in terms of the dimension of the ambient space this is actually a short vector and this is contained in the lattice that the adversary knows. So now the adversary can try to use L cube to approximate short vector and hopefully get f g but what how much time will L cube take?

(Refer Slide Time: 20:41)



But using L cube algorithm to find it takes time greater than 2N to the 6 right remember m to these 6 and L square so bit size here is if you look at the matrix you have q in so many places. So that is log q bit size of the constants right which is what 2N is around 500 and log q, q was 1 twenty so this was 7 square. So this is around 10 raise to 18 which is impractical. So L cube algorithm will be impractical even if the L cube algorithm finds f g it is a very impractical way for the adversary.

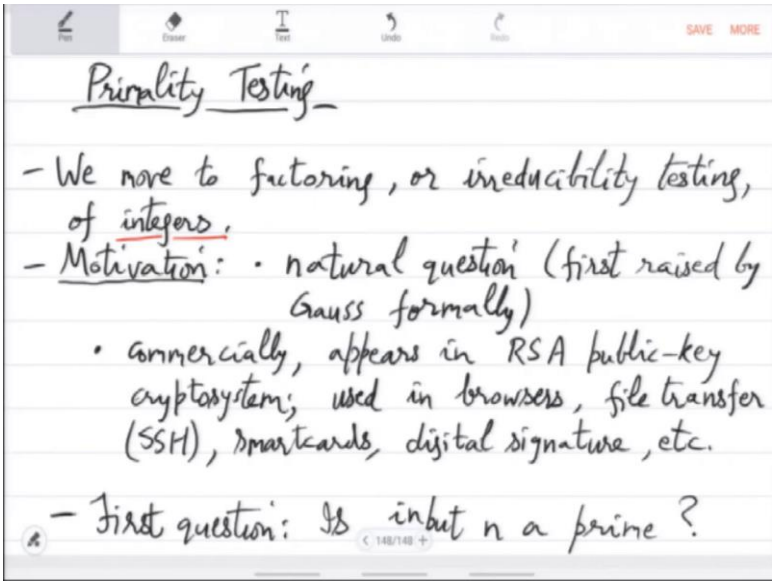
So the scheme is secure NTRU is secure for take q to be around N so in practice it is taken between N by 3 to 2N by 3, N is taken to be at least 251. This support bounds are taken to be at least 80. So note that if you want if the adversary wants to guess if it is very hard. So guessing f directly takes around 2 raised to so how many plus minus 1's are there in f they are 80 times 2 roughly right so 2 raise to 160 which is huge its completely infeasible.

So that is it so this so for even for N like 251 this is called NTRU 251. This is the starting point and then as you increase and it becomes a very secure practically secure, crypto system

which is secure even assuming quantum computation. So that finishes lattice based methods part of the course right. So we did finish NTRU and we learnt how to find reduced basis and by reduced basis we learned how to factor integral polynomials right.

So, that is those are the fundamental things that you have learnt and these are very important techniques right.

(Refer Slide Time: 25:17)



So now we will start a completely new topic we instead of polynomials we have been doing polynomials till now. Now we will move to numbers and first question we will ask is whether a number is irreducible which is the same as asking; whether the number is prime. So we will start with Primality testing so whether a given number is prime if it is not then you will you would want to find factors which is the question of integer factoring.

So in the next classes we will finish these 2 topics integer factoring; Primality testing we will find fast algorithms very fast algorithms integer factoring is still an open question in practice and in theory based on the hardness of integer factoring there is this other public key cryptosystem which is kind of more famous and used everywhere called RSA. So we will also cover that. So we move to factoring or irreducibility testing of integers.

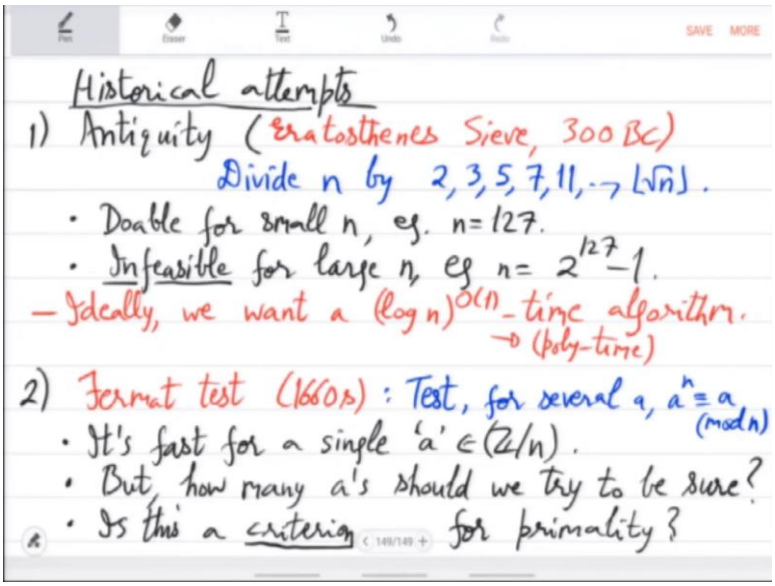
So till now we have seen factoring and irreducibility testing of polynomials that was what we saw in this course till now. Now we move to integers. And these questions in a way are harder they need more specialized techniques than what you have seen till now. So what is the motivation for studying integers? Factoring questions irreducibility questions. So this is a

very natural question first of all probably first raised by Gauss formally whether you can test numbers for Primality and then factor.

Second more recent reason is commercially these questions appear in RSA cryptosystem public key crypto system which is used in almost everywhere on the internet where you want security. So it is used in browsers, file transfer, like SSH, smart cards, digital signatures etcetera. So it is both a fundamental mathematical question and it is also used in the industry. So by it I mean both primarily testing and integer.

So let us start with primalty testing. Is input N a prime? And this input will be given obviously in binary. So you are given login bits so for l you bits able to represent a 2 raised to a large number. So for 100 bits you are representing something as big as 2 raised to 100 and for 1000 bits something as big as 2 is to 1000. So you are actually working with extremely large numbers and you are interested in their Primality.

(Refer Slide Time: 30:54)



So, what is the history behind this question. So very old methods so from the antiquity so Eratosthenes Sieve from 300 BC this is essentially by the definition of prime numbers right. So what you do is the algorithm is just divide N by 2, 3, 5. So primes up to you only need to go up to square root N. So divide N by the primes appearing before square root N and if it is not divisible by any of these then it means that N is a prime number that is the algorithm.

So this is doable for small N example N equal to 127 this is a good way to work by hand and find whether 127 is prime or not. But this is infeasible even impossible for large N example

what if N is 2 raised to 127 – 1, 127 bits. So now the number is so large that its square root is something like 2 raised to 64 or 63 and 2 raised to 63 steps is a lot of steps this becomes actually impossible in practice.

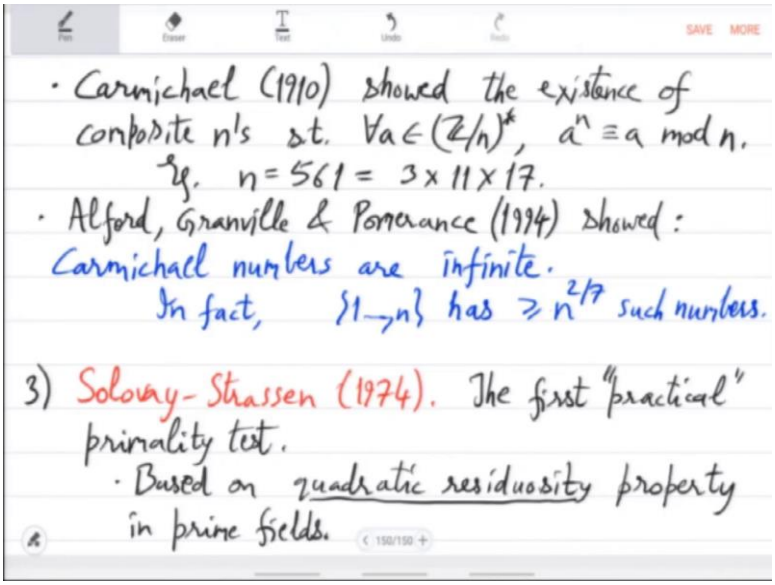
So those were the methods known classically. So, complexity wise what we want is ideally we want a login to the constant time algorithm with so this will be a polynomial time algorithm. This will be a polynomial time algorithm that is what we want but this method the previous method square root N steps is something like 2 raised to log N. So it is exponential time.

So let us move to the next step or next slightly more advanced primality test which was discovered it is called Fermat test which is fairly recent in terms of with respect to 300 BC this is very recent 1600s 17th century. So what is done is this uses Fermat's little theorem. So you know that if N is prime then a to the n is a mod N right the Frobenius map. So why do not you test that.

So test for several a whether a to the n is a mod n if it is then you give up and say that n is prime if it is not then you are 100 sure that n is composite and cannot be prime. So it is fast for a single a picked from z mod n but how many a should we try. So can we just randomly pick an a and b confident that N is prime or do we have to check for all a. Now even if we check for all a maybe the test passes for all a but n is not prime right.

This may not be a criteria or criterion for Primality testing. So is this a Primality criteria. So it was shown later much later it was discovered that this is not a criterion even composites satisfy Fermat’s little theorem.

(Refer Slide Time: 38:03)



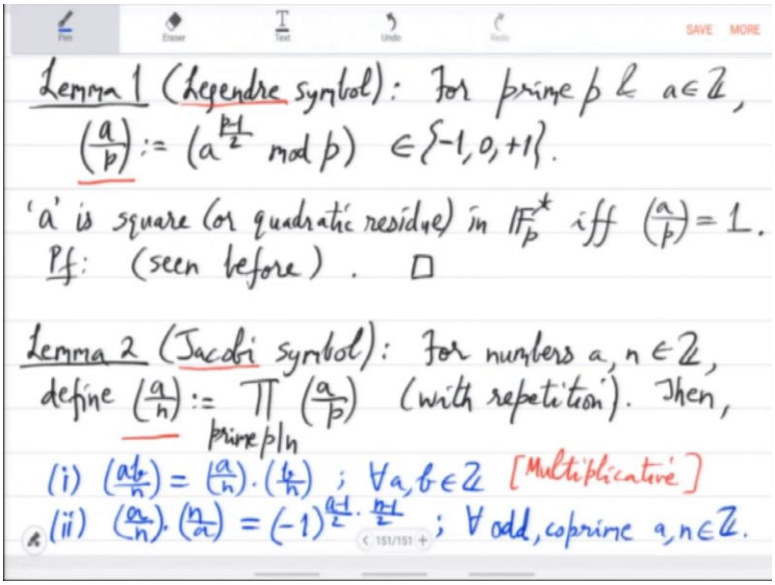
So Carmichael showed the existence of composite ends such that for all a co prime to n for all a co prime to n e to the n is a modern which shatters our dream of using this idea of Fermat theorem for in Primality testing because even composite n can pass this no matter how many a's you pick. So the example of smallest n is 561 right so this is 3 times 11 times 17 it is clearly composite. It is also square free because all the factors none of the factors repeat.

And you can check that for all the co prime is a to the 561 is a mod 561. In fact situation was even worse or is even worse. So Alfred Granville and Pomeranze they showed that such ends are infinitely many. They showed that Carmichael numbers are infinitely many in fact they also gave a density lower bound. So in fact in 1 to n has at least n to the 2 by 7 such numbers. So they convincingly show that Fermat's test fails horribly.

There are a lot of counter examples even if you try all these so let us move to the first achievement where mathematicians found a test that correctly works and identifies prime numbers. So this is by Soloway and Strassen as recent as 1974. So that is the first practical Primality test it is based on quadratic residuocity. Quadratic residuosity property in prime fields so this property you have already seen we covered this.

When we were trying to factor univariate polynomials modulo or prime. So there you saw how to test whether a given number is a square mod p right in particular we showed that the number of squares and the number of non squares mod p is equal exactly equal its p - 1 by 2. So we want to make it into a test Primality test.

(Refer Slide Time: 44:11)



So let us recall that we what we call Legendre symbol. So for prime p and an a integral I think we saw this a by p login symbol is e raised to p - 1 by 2 mod p this value is either - 1 zero or +1. If a is 0 it is 0 but if a is a square then this is 1 if it is not a square it is -1 this is called Legendre symbol and the property is a is square or quadratic residue in f p star if and only if a by p is 1.

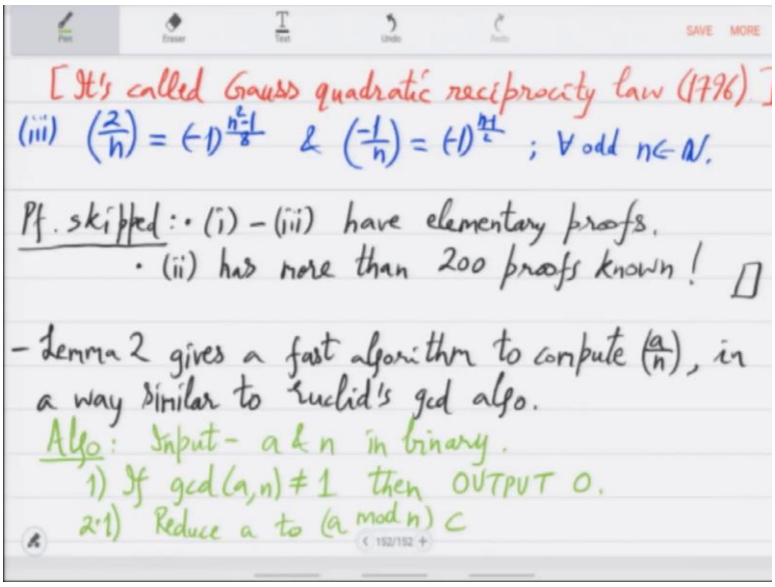
So, let us ignore a equal to 0 case. For all the other quadratic residues a by p is 1 and if a by b is 1 then a is a quadratic residue right this proof you have seen before. Now what is the analog of this when p is not a prime. So note that this a by p you can compute given p a prime but in the case when piece composite first of all how do you define the analog and second whether you can compute it.

So the analog is called Jacobi symbol. So for numbers a and n, n a by n is defined to be factorize n into primes and then multiply the Legendre symbol and with repetition. So if p square divides a then you will be multiplying a by p 2 times. So this is called Jacobi symbol and this has amazing properties. So what are the properties? So the first property is that it is multiplicative.

So ab by n is equal to a by n times p by n for all ab integral. Second property is what is called let me name it so this first one is multiplicativity. Second one is what is called reciprocity law. What it means is that if you know a by n, n by a is correlated. So a by n times n by a is just a sign and we also can actually find out the sign easily this is - 1 to the a - 1 by 2 times n

- 1 by 2 that is the sign for all odd and co prime a, n. So this is called gauss's quadratic reciprocity law.

(Refer Slide Time: 50:43)



Gauss proved it in 1796. So these are the 2 major properties of Jacobi symbol it is multiplicative and it satisfies the reciprocity law. Third property is it is basically based on when is 2r square modulo prime and when is - 1r square modulo prime. So you can work it out and you will find that 2 by n is -1 to the n square - 1 by 8 and - 1 by n is - 1 to the n - 1 by 2 for any odd n.

So this is a minor calculation but the major properties were this multiplicativity and quadratic reciprocity. We will not give the proofs of this this is standard this can be found in standard number theory books. Proof we have skipped just one remark is so 1 and 2 they have elementary proofs in fact all of them have elementary proofs which you can find in textbooks.

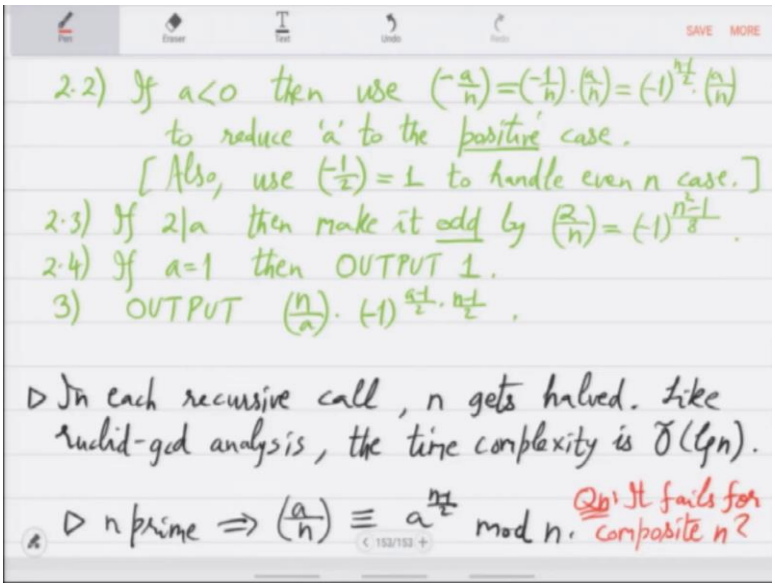
Especially this second property the reciprocity law this has more than more than 200 proofs known. So this is such a fundamental law of numbers that different mathematicians have given different viewpoints. So, very different proofs are known. So now this lemma these 3 properties in blue that I have written these are very important for us because using this we can compute a by n for given a and n numbers.

And once we can compute that we can actually describe Solvay Strazan's test. So let us do that. Lemma 2 gives a fast algorithm to compute a by n, in a way similar to Euclid's gcd. So this so that may sound surprising but the key thing will be this reciprocity law. So reciprocity

law will give you an algorithm that syntactically looks similar to Euclid gcd. So let us write down the algorithm.

So in the input you are given a and n in binary. So if compute the gcd of this. If the gcd is not 1 then you know that the Jacobi symbol which actually is defined via the Legendre symbol it will be 0. So we will assume that a and n are co-prime and moreover the interpretation of a is mod n right. So reduce a modulo n between -n by 2 + n by 2. So do this reduction.

(Refer Slide Time: 57:13)



And now there are 2 cases. So a may be negative or may be positive. Suppose a is negative then use the fact that - a by n is equal to -1 by n times a by n and - 1 by n is - 1 to the n - 1 by 2 times a by n. Here we use the fact that n is odd let me first finish this line of thought. So if a is negative then use this to make a positive. So you can make a positive by switching a to -a

by this formula but this formula requires n to be odd.

If it is not odd also use you can actually use -1 by 2, -1 by 2 is 1 to handle the even n case. So basically remove the extra 2 power in n using this formula again and again - 1 by 2 and then when you have an odd n you use the – a, a relationship. So this way you will make a positive. So we can now assume a to be positive without loss of generality next is let us make a odd. So if 2, divides a then make it odd by using the fact that 2 by n is - 1 to the n square - 1 by 8. So, this allows 2 to be taken out right again n can be assumed to be odd and then 2 by n you can use the property that is -1 to the n square - 1 by 8. So we have made a odd and is positive and odd suppose it is 1. If it is 1 then you know that 1 by n is 1 so output 1. So now you know

that a is odd positive and at least 3 what do you do now? So for example what do you do with 3 by n, how do you compute it?

So here you use quadratic reciprocity. So you instead of computing a by n you compute n by a and then the sign change could be this. Fine so the advantage now is that a has becomes around half of n and when you switch when you swap them with n will be reduced now modulo a. So n will actually now become half of a. So in every step we are having just like Euclid gcd. So in each recursive call n gets halved.

So like Euclid gcd analysis that we did in the very first week right. You so recall Euclid gcd analysis the time taken by this algorithm is only linear. So this is an o tilde login time algorithm to compute the Jacobi symbol. So Jacobi symbol can be computed very fast this is what we have learnt. And the other thing to note is that if n is prime then a by n is a to the n - 1 by 2 mod n. So Jacobi symbol is the same as Legendre symbol which is the same as a to the n - 1 by 2 mod n in case n is prime.

Now this may not be true for composite n. So does it feel for composite n if it fails then this is a criterion for Primality and you can use this to test n for Primality. So next time we will investigate this, thank you.