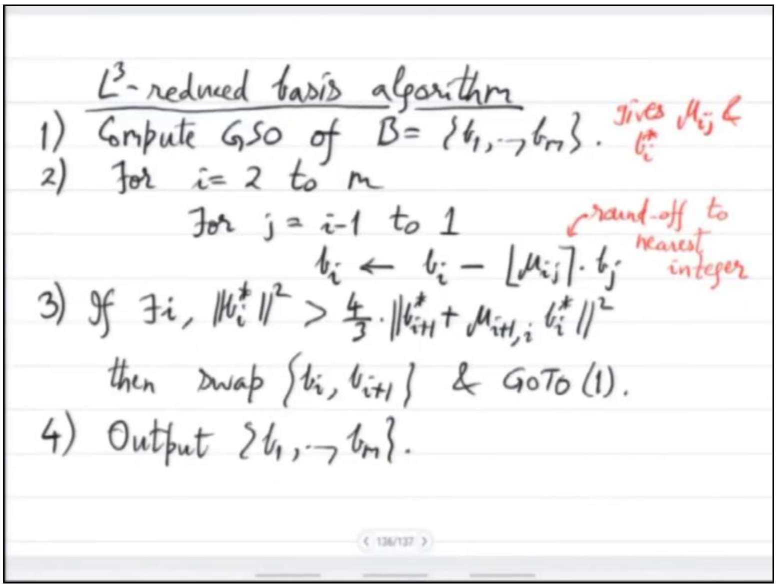
Computational Number Theory and Algebra
Prof. Nitin Saxena
Department of Aerospace Engineering
Indian Institute of Technology - Kanpur

Lecture – 22
Analysis of LLL - Reduced Basis Algorithm and Introduction to NTRU Cryptosystem

(Refer Slide Time: 00:21)



So last time we wrote this algorithm which is L cubes reduced basis algorithm. So what it does is that b 1 to b m are the linearly independent vectors that span a given lattice the your goal is to find an approximation to a shortest vector by the lattice generated by these vectors b 1 to b m. So first in step 1 you compute the Gram Schmidt orthogonalization of b. Once you so the output of that will be b 1 star to b m star.

And then the relative these projection constants would be mu ij’s. So these 2 are the standard output the data which you get out of GSO computation and then what is done is this mu ij is rounded off to the nearest integer because mu ij may be a real number it may not be an integer. So this computation you actually cannot do by remaining within a lattice so and you really want to remain in the lattice generated by b 1 to b m you do not want to go out you do not want to change the lattice.

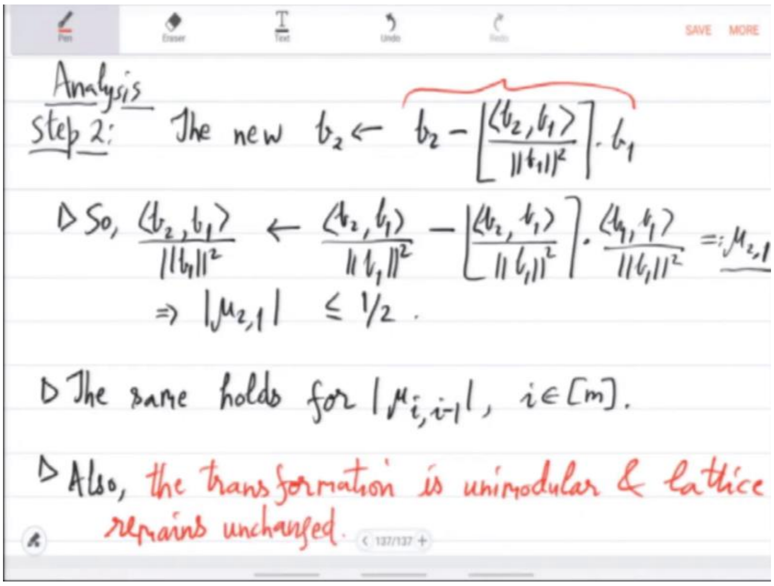
So instead of mu ij you will calculate the nearest integer and then you will only use an approximation of mu ij. So that will be the new vectors b 1 to b m they still generate the same lattice. Now that was the first condition for reduced basis. The second condition was that if you recall the second condition of reduced basis was this that is written in blue that sorry that is written in black point number one.

So point number 1 wants essentially this c 1 star to c m star in sort of an increasing order right because you want to output in the end c 1 star. So ideally c 1 star should be smaller than c 2 star should be smaller than c 3 star and so on that may not be achievable efficiently so there is an approximation here. You want c i star to be smaller than 2 over square root 3 of c 2 star, c 1 to be smaller than 2 by square root 3 of let us say c 2 then c 2 to be smaller than 2 by square root 3 of c 3 and so on.

So it will be roughly in an ascending order now if this condition is violated till now when you come to step 3 suppose this condition is violated by some b i meaning that b i star's length is bigger than this bound on b i + 1. So if this happens then the simple correction is swap them. So b i + 1, b i did not satisfy the condition but maybe when you swap b i + 1 will satisfy the condition with respect to the next.

But then since you have changed the order of the vectors you have to recompute GSO so go to step 1 and keep doing this hopefully this will terminate after finite number of steps and then you will output b1 to b m. So the analysis of this algorithm is key first of all why will this algorithm stop if it stops in how many steps and when it stops is b 1 the correct approximation of shortest vector right those are the questions to be answered yet.

(Refer Slide Time: 04:22)

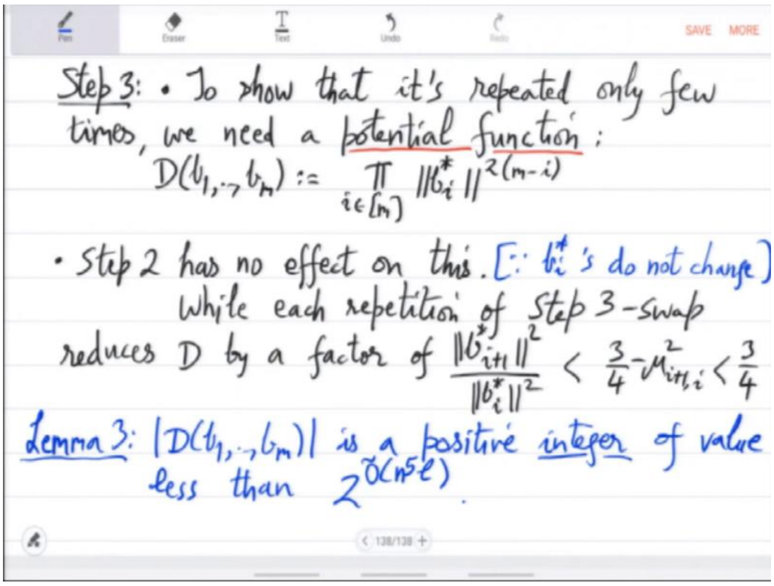


So let us start with the analysis. So step 2 step 1 is just GSO that you understand that is standard. So what happened in step 2 in step 2 we recalculated or we changed b 1 to b m following GSO but remaining in the lattice right. So the new b 2 for example is b 2 minus this projection constant rounded off times b 1 right. This is the new b 2. So note that if you look at the new projection constant b 2 b 1 right this has become, so this is b 2 b 1 divided by norm of b 1 square - b 2 b 1 norm of p 1 square times b 1, 1 divided by norm of p 1 square but this last quantity is 1 right.

So you see that you are actually subtracting b 2 b 1 over b 1 square the old 1 from that you are subtracting the nearest integer. So the difference will be smaller than half. So this value or let me be precise let me call this new mu 2 1. So mu 2 1 its absolute value so it is between minus half and half the absolute value is less than equal to half this you can check and so on. So what you have ensured is that mu ij step 2 ensures that mu ij is at most half in magnitude.

For mu i, i - 1 so this property is true for all i by the same argument also the transformation that you are doing is this right. This is the transformation now this transformation is just subtracting a multiple of b 1 from b 2 it is not scaling b 2. So since it is not scaling b 2 this is unimodular and the lattice does not change. So the lattice remains unchanged after this transformation so step 2 is good we understand that what is step 3.

(Refer Slide Time: 09:07)



Now in step 3 what you did is if this reduced basis condition is violated. So this LHS is bigger than RHS then you will swap the 2 vectors and you will repeat the algorithm go to step 1. So to show that this number of repetitions is bounded that it is not repeated to show that it is repeated few times we need a potential function. So what will be the potential function potential function should be such that to begin with it is small.

And with every swap and repetition it reduces. So we will use the following potential function. It will be the product of the lens of b i star but remember you want this thing to reduce with every swap. So intuitively what was happening bi star some bi star was larger than b i + 1 star intuitively right. So when you swap for this to be to be reducing significantly let us give b i star and b i + 1 star different weights.

So for that we put an exponent here. So it is actually b i star will get a larger exponent than b i + 1 star. So when you will swap this value will actually fall how by how much? So we will now calculate that. So step 2 has no effect on this why so step 2 was mimicking GSO and this potential function actually looks at b i star. So these axes these orthogonal axis do not change with GSO. So let us remember that.

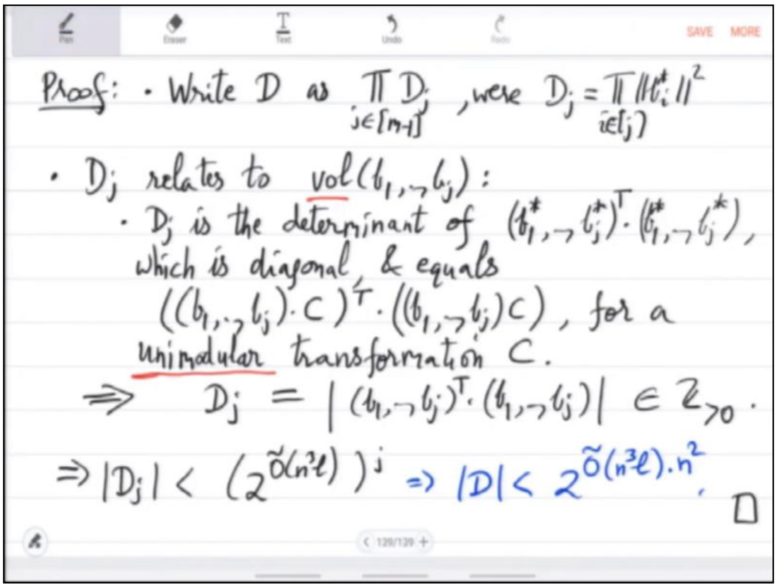
So step 2 has no effect on this. While each step 3 repetition of step 3 swap reduces D by a factor of so b i star is being replaced by b i + 1 star and bi star has more weight right. So the reduction is by this factor. And now by the reduced basis condition violation you know that b i star was actually bigger. So this is less than 3 by 4 - mu i + 1 i square now mu square is at most 1 fourth so this is this is less than what?

This is less than half sorry no mu square is no it is non-negative so this is actually less than three fourth even. So D will with every swap D will reduce by a factor of at least three-fourths what else so what was D to begin with and more importantly was D an integer or a non-integer? See if it is an integer then you know that if at every after every swap it remains an integer then you know that you can only hit one you cannot the potential function cannot go below that and then you will get a bound.

If it is a real number then it can keep on reducing infinitely many times. So we will actually show that at any point of time D is an integer. So that is an important property of the potential function. So the magnitude of D is a positive integer of value less than 2 raised to o tilde n raised to 5 l and was the ambient dimension that you started with of the lattice or the vectors l was the coefficients the coordinates in the vectors were l bit.

So we have a bound actually n is to 5 times l. This is the bit size bound on the potential function and it is an integer. Since every time you are reducing it by 3 4th this gives you number of times n raised to 5 times l. So that will be the immediate consequence of this lemma. So why is this true let us prove it.

(Refer Slide Time: 17:02)



So write D as product of D j 1 to m - 1 where D j is simply a product of b i star lengths. So instead of these exponents that we put we are removing the exponents essentially. So we are just multiplying b 1 star length 2 up to b j star length squaring it that is D j and you can see that D is equal to product of D j, D j has a nice interpretation. If you think in terms of orthogonal axis if you multiply the length of the axis this represents some kind of a volume.

So the potential function D is actually a product of volumes of sub let us say sub cubes or sub parallel pipettes. So D j relates to the volume of b 1 to b j because D j is the determinant is the determinant of b 1 star to b j star. So this is a sequence of column vectors transposed. So they become rho multiply it with this. So D j is actually the determinant of this matrix this matrix is just remember that b 1 star to b j star are orthogonal right.

So b 1 star inner product b 1 star will be the length of b 1 star square but other inner products will be 0. So this is actually a diagonal matrix and the determinant is just the product of b i star square which is equal to which is a diagonal matrix and equals. Now remember that b 1 star to b j star are computed by GSO. So they only depend on b 1 to b j right so b 1 star to b j star are just a transformation on b 1 to b j so we can rewrite it as b 1 to b j times a matrix transformation and itself right.

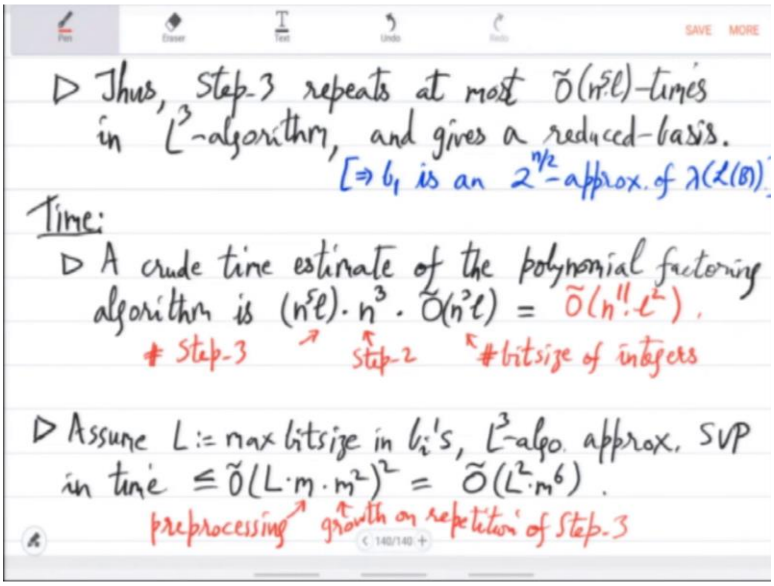
C is the unimodular transformation. So this is a nice transformation happening in the lattice well not quite but it is a basically it is a transformation GSO is a transformation such that determinant of c is + - 1. So if you look at the at the determinant version of this then you will get that D j equals so determinant of c and determinant of c transpose is + - 1. So the product is 1 right so ultimately it is the determinant of b 1 to b j transpose times b 1 to b j determinant of this matrix square matrix which is now remember that b 1 to b j are given in the input.

So they are integral entries. So the determinant of this integral matrix is integral. Moreover since the vectors were linearly independent this will be this cannot be 0 we want to b j are linearly independent right so the volume will come out to be a positive integer and the bound follows from just look at the entries of b j they are L bits so you will get actually for this you have to go back to the first algorithm.

How did you come from polynomial factoring integral polynomial factoring to this b 1 to b m instance of SVP right. So there was this n q l times hansel lifting that we did. So the integers after that will potentially grow to 2 raise to o tilde n cube l. So these are this is the magnitude of the numbers integers you have in vectors b 1 to b j. And then when you take determinant you will be multiplying these j times so you get times raised to j with and j remember is at most m which is at most n.

So this implies that D the potential function is less than 2 raised to o tilde n cube l times so you just multiply for over j going from 1 to n so it gives you another n square so that is a bound that is a lazy bound on the potential function which proves the lemma. So the potential function is a positive integer of value less than 2 raised to o tilde n to the 5l.

(Refer Slide Time: 25:20)



So now we have what we wanted we are nearing the end of the proof. So thus step 3 repeats at most o tilde n to the 5 times l times in l cube algorithm and gives the reduce basis. In fact reduce basis may be of may be many so you will get our reduce basis and why is that well you showed how many times step 3 will repeat. Once you once step 3 does not repeat it means that both the axioms of reduced basis in the definition are satisfied so then you will output the reduced basis.

So the basically the vectors b 1 to b m are now approximately ascending order and the other condition was that if you look at the projection constants they are around half. So this is like an approximation to GSO. They are kind of pseudo orthogonal. So you will just output so you can pick b 1 in the output that is a SVP approximation. So this we have done last time right last time we showed that c 1 estimates the shortest vector by this factor, we have shown this.

So just recall this identity or this inequality that c 1 star will be a 2 raise to m by 2 factor. So b 1 is and 2 raise to n by 2 approximation of the shortest vector. So we have everything that we wanted finishes l cube let us see the time complexity. So just a naive time estimator crude time estimate of the polynomial factoring algorithm is n raised to 5l. So you have to start from the very beginning.

How did you do Hensal lifting and then you reached this lattice instance and then you applied L cube algorithm. So this will be n raised to 5l these are how big the numbers are this actually is the number of times you are repeating the L cube algorithm. The first steps you are repeating and is to 5l times. What happens in this step 2 is that this is already a for loop nested for loop and when you are doing this difference b i - b j remember there are n coordinates right.

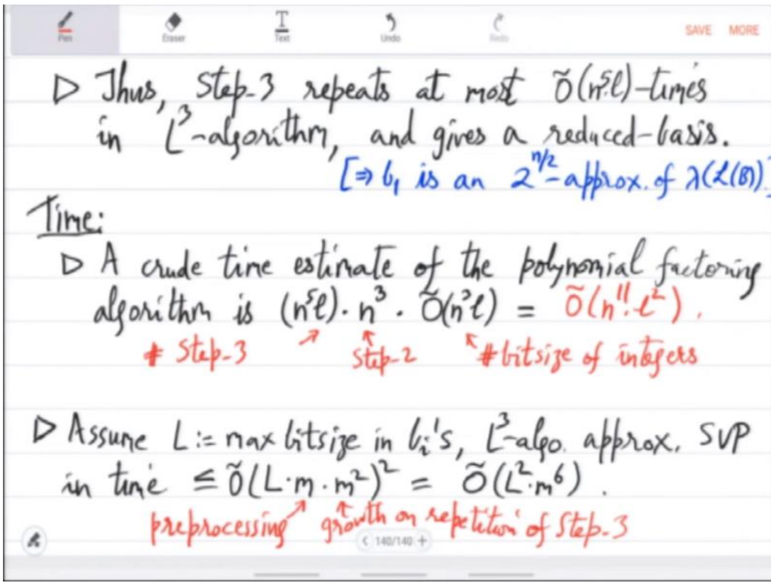
So overall step 2 takes n cube naively a naive analysis so that is another n cube and how big are these integers on which you are computing that is around n cube L. So you get o tilde n raised to 11 L square. So this is a horrible time estimate it is n raised to 11 times l square. So as the dimension increases this algorithm is extremely slow you can optimize some parts of it but you cannot make it very fast at least not theoretically.

But on practical instances it is known to work well so this is used in practice some version of this. So this comes from the number of repetitions this n cube is step 2 and this is the size of bit size of integers. So that is the time complexity of integral polynomial factoring. And if you just look at L cube algorithm so to find reduce basis what is the reduced basis finding algorithms complexity. So assume big L to be the max bit size in bit size L cube algorithm approximates in how much time?

So this will be bit size of the integers times m and every vector there m of these or this actually is of node not like that this is m is for pre-processing times m square whole square. So you get L square m to the 6. So let me give an explanation for this so this is the this is the bit size but this m we are putting for pre-processing and this m square you will get because of the number of time step 3 will repeat this comes from the potential function.

So you can think in terms of the bit size you started with L bit size when you did the pre-processing you had a growth of multiplier m and when you started repeating step 3 there was another blow up in the bit size by m square. So this is what the integers this integers are this big and so that also tells you how many times you have to repeat as step three. So you square it so this is a crude estimate in terms of m it is taking m to the sixth time.

(Refer Slide Time: 34:35)



So let me finish L cube algorithm by giving some properties that it satisfies its output satisfies so L cube algorithm and reduced basis is used in many places. So examples are computational problems in algebraic number theory, faster arithmetic in number field's knapsack problem and so on. So the key properties of this reduced basis which are useful there in these applications is the following. So let me collect that here so let b 1 to b m be a reduced basis of the lattice L which is in the ambient space let us say z to the n and b 1 star to b m star be the GSO.

After GSO you get b 1 star to b m star on this then the things which are satisfied and these properties you can deduce from the calculation that we just did the calculations we did in the last 2 lectures using that you can reduce these properties. First property in fact we deduced it is it says that the length of b j never exceeds the length of; so b i - 1 let us say j equal to b i – 1, b i - 1 is at most b i star times 2 raised to i - 1 by 2 for all j behind i.

So in particular it is saying that if you look at b i – 1 it is a factor smaller than b i star and this factor is exponential in i 2 raised to i by 2. This can be shown by this inequality axiom this almost ascending property of the reduced basis it follows directly we did that calculation. Second is if you look at the product of b i's length I think for this I needed n, n would be simpler 1 to b n, b n star to b n star there is no m or m is equal to n.

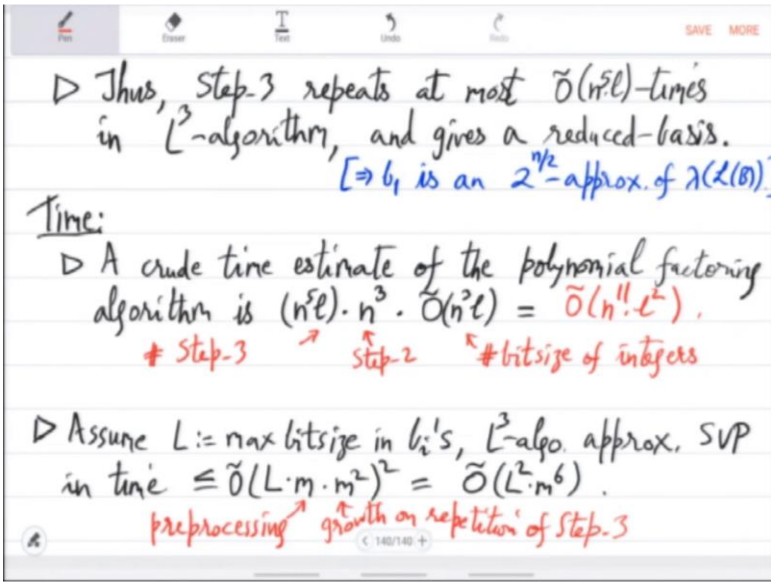
So this product of b i of the reduced basis i if they were if b i's were orthogonal this would have been the volume of the lattice right. The volume of the lattice is the same as the determinant. So the product of these b i's is at least the determinant of the lattice and it is at most this multiple 2 raised to n n - 1 by 4 times D L in other words if you look at the product of the lengths of the vectors in the reduced basis.

This is an approximation of the determinant it is a 2 raise to n square approximation this is basically intuitively what it is saying is that the b1 to b n that you get are pseudo orthogonal. So their product also is not very far from the volume. Third is that b 1 right which is the vector you pick an output as an approximation of SVP this is then related to the volume like this. So one can be obtained can be proved inequality one can be proved by the inequality axiom of reduced basis.

From that it easily follows that product of b i is related to the determinant and from the determinant then you can deduce something about b 1 because you know the relative ratios of b i's b i lens so from that you will get actually that b 1 is basically it is smaller than in step 2 in RHS look at 1 by nth power. At this point I would like to remark that the shortest vector in the lattice has length around square root of n times the volume by raise to 1 by n.

This is a result by Minkowski this is a classical result that shortest vector in lattice has length around the volume raised to 1 by n times square root n. So if you look at b 1 that you are outputting this is close to that it is only off by 2 raised to n by 4. So again this is an exponential approximation but theoretically it is connected to many things. So I will not go into the proofs of each of these I leave it as an as an exercise.

(Refer Slide Time: 44:15)



So one you prove by use this condition in the reduced basis. Condition number one reduce basis definition for the second one this relationship with the discriminant just use the above and the fact that the discriminant or volume is product of it is the determinant of this matrix which is then less than equal to product of the lens. And then you can use again inequality one which will relate b i star with b i.

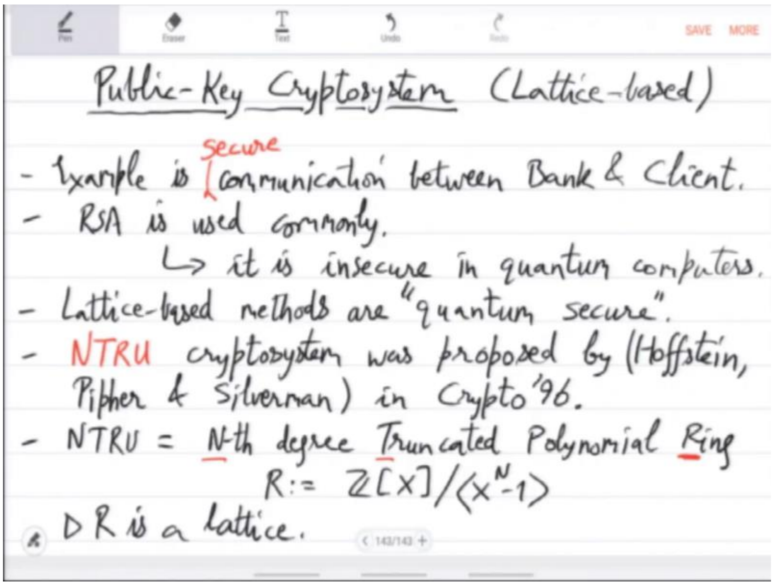
So that is how you will get inequality 2. Inequality 3 will follow by the ratios of p i's. So by 1 you get that b 1 length is less than equal to 2 raise to -1 by 2 times p i star right this is what inequality 1 gives you for all i. So using this and inequality 2 that you got you will get 3 and the fact that product of bi star over all the i's is product of b i star square is dL yeah that is true because b i stars are orthogonal.

So I should have made this inequality actually that is true which is also the volume of the lattice. So using the same fact and this inequality 1 will give you inequality 3. So these are the 3 properties why which are always invoked in the applications major applications. So one application where I can stop now is given rationales alpha 1 to alpha n and epsilon find integers p 1 to p n q such that p i by q approximates alpha and minimize q.

So you are given these rationales alpha 1 to alpha n and you want a rational representation for all these alpha is the catch is that denominator should be the same. So this problem is called simultaneous Diophantine approximation. And Diophantine because you are given these rationales let us say alpha 1 and you want to write it as p 1 over q and simultaneous because you are given alpha 1 alpha 2 and you have to simultaneously express it as p 1 by q and p 2 by q.

Now obviously q exists but then it could be very large if you do it naively. You want to minimize q as much as possible and this problem surprisingly could be solved in polynomial time using a reduced basis L cube algorithm. So this also I will leave as an exercise. So solve it by L cube reduced basis algorithm. So with this I will stop the discussion on L cube algorithm and I will move to an application in security.

(Refer Slide Time: 51:38)



So, we will look at a public key crypto cryptosystem which will be lattice based. So what is a public key crypto system so this is a mechanism that is used in a number of situations. One simple example is when you interact with your bank. So bank has given certain information to the public using which you can send your private information to the bank which only the bank can understand nobody else could understand over the internet.

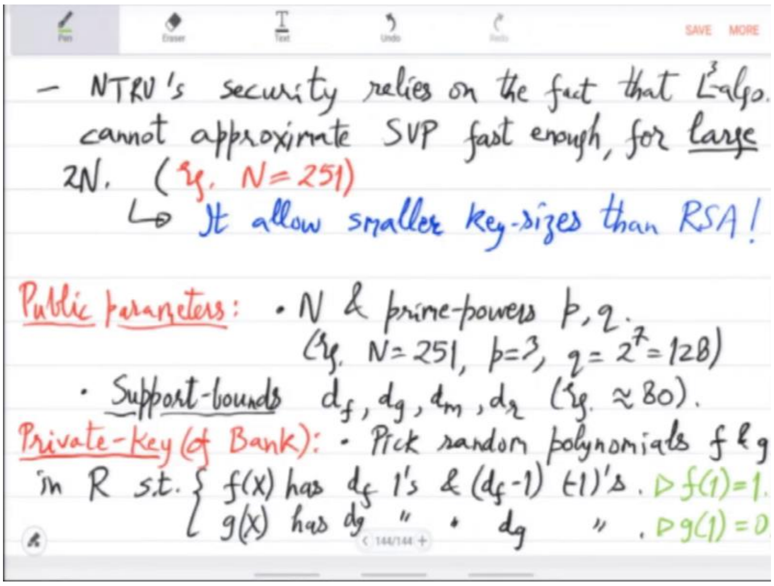
So let us fix that example. Example is communication between bank and client and obviously it should be secure, secure communication. So one public key crypto system which is used quite commonly is RSA but the problem is that it is insecure if quantum computers exist. Now in the future quantum computers may exist. So people believe that we should have a mechanism which can even which can work even if quantum computers are created in the future.

So for that lattice based methods are used this is why this is a major application of the lattices. Not only security in quantum computers but there are also other advantages for using lattice based cryptography. So lattice based methods are quantum secure so there are many systems proposed we will see here the basics of NTRU cryptosystem. So N T R U was proposed by Hofstein, Pipher and Silverman in the conference crypto in 96.

And this basic protocol is very popular and there are a lot of variants of this known some of which are also implemented and they work well. So NTRU stands for Nth degree truncated Polynomial Ring. So NTRU so what is this polynomial ring? This is this is very simple it is just integral polynomial ring mod out Z x by x to the N – 1. So this is a lattice it's an n-dimensional lattice and all the computations will happen here.

So the message the private key the public key they will all elements of this ring. So they will all be lattice elements and we will do little computations with them simple computations.

(Refer Slide Time: 57:47)



So NTRU's security relies on the fact that the L cube algorithm cannot approximate SVP fast enough for large N. So maybe I should say for large to N and N will be taken let us take N to be 251. So N is 251 means that 2N is 500 and we have seen that the complexity of L cube reduced basis algorithm is something like 500 to the 6 times other factors and that in practice actually will be very slow and as you increase n it will get harder.

The L cube algorithm will become slower. So, one advantage of this is that it allows smaller key sizes than RSA. So, to get the same security same level of security RSA would require thousands of bits but here you can do in 200, 300, 400 bits. So the key sizes will also be small so there are many advantages in this system. Let us quickly go through the public key private key encryption and decryption.

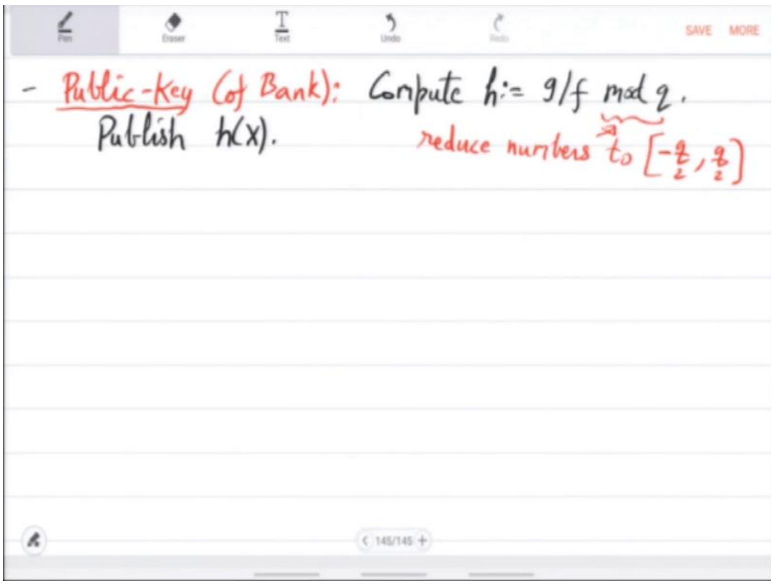
So what are the public parameters that the bank will release? So bank releases N and prime powers p q so you can keep this example in mind N, p, q N is 251 p is 3 q is 2 raised to 7. So 128 and the bank releases support bounds. So these are just numbers we will call them d f, d g, d m, d r these are around again example numbers around 80. So they are not very large. So this is what the bank fixes and also publishes them on the website.

What is the private key? So, private key of the bank what does the bank keep private and will not publish. So this is pick random polynomials f and g in R such that so f has exactly d f 1's and 1 less -1's g x on the other hand has equal 1's and -1's. So again f has d f which is the parameter published by the bank that many ones and minus ones are 1 less. So this means this immediately means that f at 1 is 1.

G on the other hand has equal number of 1's and -1s so g s at 1 is 0. So randomly these polynomials are picked there is a lot of choice because if you take d f, d g to be 80 then there are around 80 positions to be picked out of N positions right. So, 80 positions out of 251 and even after picking the positions you still have to decide on plus minus 1 sign. So there is a lot of there are at least I would say 2 raise to 80 at least 2 waste to 80 possibilities are there.

And bank picks 1 of them and keeps it private and what is the public key which the bank releases.

(Refer Slide Time: 01:05:04)



So public key it come; uses f and g to compute g by f. So compute g by f mod q and publish this. This one thing I would like to say here is that this mod q computation you should reduce numbers to + - q by 2. So you have coefficients between -q by 2 to + q by 2 that is your hx bank publishes it. Note that from h it is not clear how to compute g or f right because h is g by f there are 2 unknowns.

So from h it is not clear how to find those 2 unknowns. So they seem to be still a secret. So next time we will see how to do encryption and decryption and talk about the security of this vis-a-vis L cube algorithm ok I will stop, thank you.