

Cryptography and Network Security

Prof. D. Mukhopadhyay

Department of computer science and Engineering

Indian Institute of Technology, Kharagpur

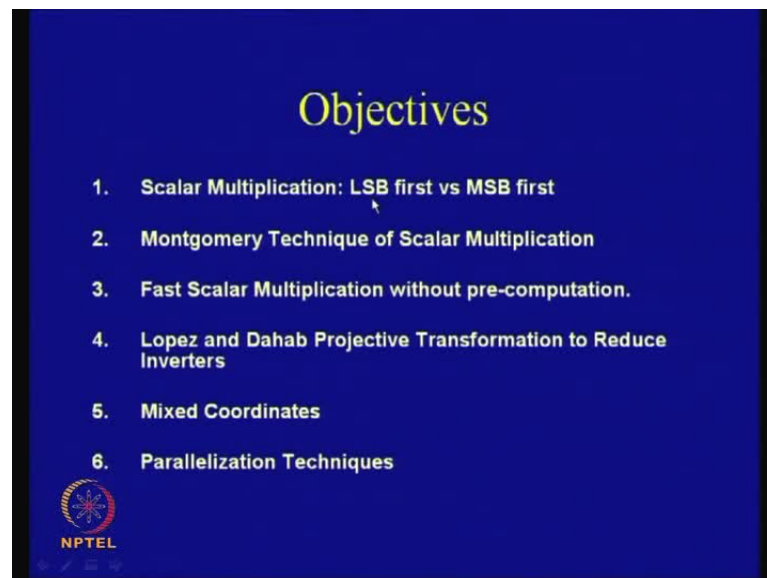
Module No. # 01

Lecture No. # 36

Implementation of Elliptic Curve Cryptography


So, in today's class we shall discuss about implementation of elliptic curve cryptography. So, as we have seen in the previous classes is that, elliptic curves and elliptic curve cryptography essentially relies upon point operations, like point addition and point doubling operations. So, they are all actually, quite **computation** or intensive operations. Therefore, if I want to develop either a software or a hardware, we need to take some more, or rather, there are some interesting developments, how you can actually implement it in much more efficiently. So, today's discussion will be based on this.

(Refer Slide Time: 00:59)



Objectives

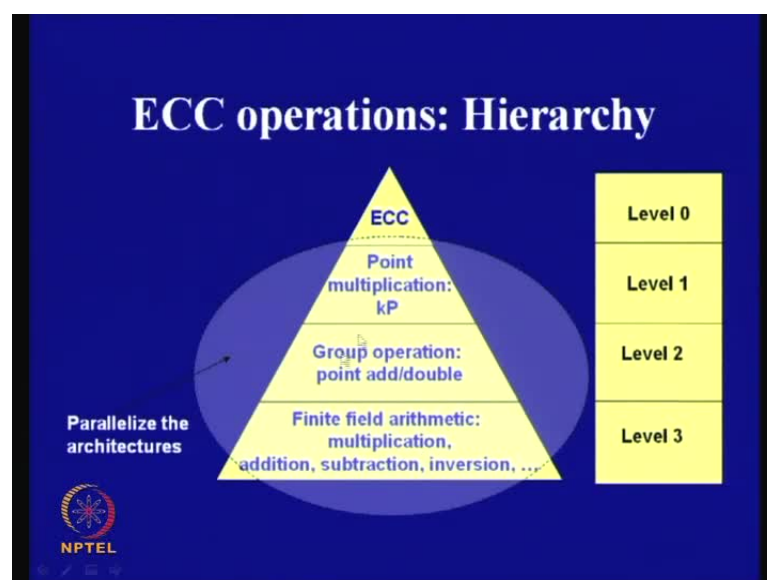
1. Scalar Multiplication: LSB first vs MSB first
2. Montgomery Technique of Scalar Multiplication
3. Fast Scalar Multiplication without pre-computation.
4. Lopez and Dahab Projective Transformation to Reduce Inverters
5. Mixed Coordinates
6. Parallelization Techniques

 NPTEL

So, we shall discuss about scalar multiplications and we shall discuss about, whether LSB first or MSB first, is approach, I mean, we will try to compare between these two approaches. Then, we will discuss about Montgomery technique of scalar multiplication and develop, discuss about fast scalar multiplications without pre-computations; like, before this, there were some people, I mean, there were some implementations, which were actually based on pre-computed values of... Suppose, I want to compute λP . So, they were like, some pre-computed values were already stored and then, they were combined to compute the value of λP . This is one approach, could be like, if I want to compute λP , then, you take the binary encoding of λ and you keep certain values of say, may be 2λ , 3λ or 4λ , 8λ stored and then, you combine them in order to obtain the value of λP .

So, they are actually, like scalar multiplications using pre-computed tables, but there may be applications, wherein, you may not be interested in spending so much amount of memory. And therefore, they were quite nice, I mean the methodized paper by Lopez and Dahab, to show how you can actually do fast scalar multiplication without pre-computation. So, we will study that. And we will study about Lopez and Dahab Projective transformation to reduce the number of inversions or finite field inversions, which are necessary; discuss about, little bit about mixed coordinates; not go much into details and rather, concentrate about some parallelization techniques, which can actually accelerate the scalar multiplication operation.

(Refer Slide Time: 02:21)



So, let us go step by step. Therefore, this is the, how the broad diagram of elliptic curve hierarchy looks like. So, you can see that, the basic elliptic curve operation is based upon point multiplication. I called it kP , that is, k is a scalar and P is a base point. And, the other one is and it is based upon group operations, which are point addition and point doubling. And, that is again, underlying, based upon the finite field operations like addition, subtraction and inversion and so on. Now, you see that, there are... That means, that, there are various levels of how this elliptic curve cryptography operation works.


So, therefore, if I want to obtain accelerators, then, I will try to parallelize this architectures, right. So, I will try to parallelize the point multiplication; I will try to parallelize the group operations; I will try to parallelize this underlying arithmetic operations. So, therefore, for, what is important is, to understand, where are the scopes of these parallelization's or how we can actually obtain these accelerations?

(Refer Slide Time: 03:22)

Scalar Multiplication: MSB first

- Require $k=(k_{m-1},k_{m-2},\dots,k_0)_2, k_{m-1}=1$
- Compute $Q=kP$
 - $Q=P$
 - For $i=m-2$ to 0
 - $Q=2Q$
 - If $k_i=1$ then
 - $Q=Q+P$
 - End if
 - End for
 - Return Q

Sequential Algorithm
Requires m point doublings and $(m-1)/2$ point additions on the average

 NPTEL

So, first, let us concentrate upon how to do this scalar multiplication operation. So, there is a point P and we are interested in computing k multiplied by P . So, k is my scalar, which can actually be written as a , can be encoded in a binary format; call it k_0, k_1 and till k_{m-1} . So, k_m is actually, the first time, when 1 is encountered; previous to that, everything was 0. Do you understand this? Therefore, if I want to encode 7, it will be 1 1 1; you want to encode as, encode 5, that it is, k is 5, then, it is 1 0 1; previous to that, it is all 0s.

So, therefore, very simple way is, what is called as the double and add algorithm. So, what we do is, that, we take this P and stored in a temporary register, call it Q and for i equal to m minus 2 to m minus 0, that, it is the first one; therefore, we take, start from m minus 2 and go till 0, and whenever we see that, we always do a doubling operation and whenever there is a 1, then we do a Q equal to Q plus P , that is the point addition operation.


So, this, in this case, you are actually parsing the scalar from MSB to the LSB. This is the least significant bit and this is the most significant bit. So, we are going from MSB to LSB; therefore, this algorithm is called MSB first. So, in this case, what is the total number of computations necessary? You see that, it requires, if there are, I mean, roughly if you just see that, it is m ; that is, the length is m ; it requires roughly m point doublings; because, you are always doing a doubling operation. It is actually for m minus 2 to 0, there it is m minus 1, but approximately I have written it, just throwing away the constant term; that is, roughly it requires, proportional to m point doublings. What about the number of additions required?

So, generally, if you assume a random k , that k will have half number of 1s and half number of 0s, if k is thought to be a, if I consider the average complexity of these algorithm. So, in that case, there will be m minus 1 by 2 point additions, which are necessary. Therefore, I need to do for very number of point additions, that is half number of point additions; that is a rough cost of this algorithm.

(Refer Slide Time: 05:50)

Example

- Compute $7P$:
 - $7 = (111)_2$
 - $7P = 2(2(P) + P) + P \Rightarrow$ 2 iterations are required
 - Principle: First double and then add (accumulate)
- Compute $6P$:
 - $6 = (110)_2$
 - $6P = 2(2(P) + P)$

 NPTEL

So, therefore, you can consider an example. If I want to compute 7 into P, I will encode 7 as 1 1 1. So, you note that, this is the first time when you have got a 1. So, when you are encoding for MSB, then, this is the first 1 that you are actually considering. If it is a 1, then, you have to do not only a 2 into P, but also add that with the corresponding P; that is, you have to do a doubling, as well as, you have to do an addition operation. And next one is again a 1. So, it means, that means, again you are doing a doubling and adding with P, that is 2 into 2P, you see that 2P plus P is 3P into 2 is 6P plus P is 7P.

So, you see that two iterations are required; the first one is double and then add, that is the principle. So, first you are doing a doubling and then, you are doing an addition or accumulation operation. So, similarly, if you want to compute 6P, it is 1 into 1 into 0. So, in this case you have to do, because of this 1, you have to do into 2 into P plus P, but since this is 0, you have to do only a doubling operation; you do not need to do the addition operation. So, in this case, you see, it is 3P into 2, which is 6P, right. So, that is the LSB first approach of obtaining the scalar multiplication.

(Refer Slide Time: 07:01)


Scalar Multiplication: LSB first

- Require $k=(k_{m-1},k_{m-2},\dots,k_0)_2, k_m=1$
- Compute $Q=kP$
 - $Q=0, R=P$
 - For $i=0$ to $m-1$
 - If $k_i=1$ then
 - $Q=Q+R$
 - End if
 - $R=2R$
 - End for
 - Return Q

Can Parallelize:

The accumulation and doubling can be stored in separate registers.

On the average $m/2$ point Additions and $m/2$ point doublings



Similar to this, you can also actually do an LSB first algorithm also; that is, where do you go from k_0 and go to the MSB. So, in this case, again, you see that k_m is 1 and these are the corresponding terms; corresponding binary values. Now, I want to compute again, Q equal to k into P . Now, for this, what we do is that, we actually, instead of having one register, we actually choose two registers; call it, call one of them as Q and the other one as R . So, what we do is that, we initialize Q to 0 and R to the value P . Now, for i equal to 0 to m minus 1, if k_i is 1, then we add onto this register Q ; that is, Q is equal to Q plus R and in the other register, we are actually doing a doubling operation.

So, here, you see that, as opposed to the previous algorithm, previous algorithm you were doing a doubling and then, you were doing an addition operation; that means, there are two steps, which you cannot actually parallelize; they are sequential steps. But in this case, because you are doing this operations on two different registers, and actually, you are actually giving more space, you are actually saving time. So, we can actually, parallelize these two operations and you can do the accumulation and the doubling in two different registers simultaneously, right. You can do this Q equal to Q plus R and in the other transistor, you can do this R equal to 2 into R operation; and this, you can repeat for all these values.

So, therefore, the accumulation and doubling can be stored in separate register. On an average there are m by 2 point additions and m by 2 point doublings which are measured.


Why so? Because, you are doing the addition and doubling in the in two different, two separate registers. Therefore, always there is a cost of m by 2 . Therefore, you see that, you are, whenever there is an addition required, so, if we assume that, half, this k has got half one terms, then, half of the times you will have, that is m by 2 times; you need an addition operation, that is m by 2 addition.

And when you do not need a addition, you need a doubling operation, right. Therefore, the total time is m by 2 point additions plus m by 2 point doubling operations. Now, why it is not m point doubling operations? Because the addition, generally, will take more time than a doubling operation. So, if you, even if you parallelize, you can assume that, the addition will actually take more time. So, you have to wait, since there are two parallel steps, you have to wait for the longest time, right. So, that is m by 2 point additions and then, you have to this rest of m by 2 point doubling operations. So, therefore, you see that, there is some advantage; because, you can actually parallelize these algorithm through this. Therefore, this is the LSB first approach of doing it.

(Refer Slide Time: 09:56)

Example

- **Compute $7P$, $7=(111)_2$, $Q=0$, $R=P$**
 - $Q=Q+R=0+P=P$, $R=2R=2P$
 - $Q=P+2P=3P$, $R=4P$
 - $Q=7P$, $R=8P$
- **Compute $6P$, $6=(110)_2$, $Q=0$, $R=P$**
 - $Q=0$, $R=2R=2P$
 - $Q=0+2P=2P$, $R=4P$
 - $Q=2P+4P=6P$, $R=8P$



So, therefore, now you see that, this is also, can be seen, this the same $7P$ and $6P$ example. So, here it is $1\ 1\ 1$; Q is equal to 0 and R is equal to P . Now, you see that in three time steps you can actually do this Q plus R and in parallel, you can do this R equal to 2 into R operation. So, because this is a 1 , now, you are doing LSB first. So, you are first starting with this one; this is a 1 . So, you are doing an addition, you are doing a


doubling. Next time, you are again getting a 1. So, you are again doing an addition, again doing a doubling operation. Next time, you are again seeing a 1; you are doing an addition, you are doing an doubling operation. So, that is the result, the register is in, the result is in Q.

Similarly, when you are doing a 1 1 0, the first thing, is 0. Therefore, 0 means, you are not doing the addition operation. So, Q is unaltered, unaffected; R is doubled. So, R becomes 2 into R. Next, you get a 1. Therefore, you add 0 with 2P, that is, you add that current value of P and the current value of R, and it becomes 2P. You double R, it becomes 4 P. Next time, you add 2P with 4 P, because it is a 1 and you double that; this is in consequential because of final result is stored here, 6P. So, therefore, you see that, you can still do the computations 7P and 6P. But in this case, you are actually parsing from the right and going to the left, right. So, you see that, there are small implications, where you are doing a, from the left to the right or from the right to the left, which you can use to your advantage, depending upon the platform you are implementing, your constants.

(Refer Slide Time: 11:31)

Compute 31P:

MSB First	$31=(11111)_2$	LSB First
<ol style="list-style-type: none"> 1. Q=2P 2. Q=3P 3. Q=6P 4. Q=7P 5. Q=14P 6. Q=15P 7. Q=30P 8. Q=31P 		<ol style="list-style-type: none"> 1. Q=P, R=2P 2. Q=3P, R=4P 3. Q=7P, R=8P 4. Q=15P, R=16P 5. Q=31P, R=32P

 NPTEL

So, therefore, one example here shows that, if I want to compute 31 P and this is an MSB first approach, and this is the LSB first approach. You see that, there is a significant reduction in terms of the time steps that are required to do this operation. But the cost that you pay is, two registers. That is the cost which you are paying. Therefore, **it in**, we

can say that, that has more scope of parallelizing, when you are doing, taking an LSB first approach.

(Refer Slide Time: 12:06)

Weierstrass Point Addition

$$y^2 + xy = x^3 + ax^2 + b, (x, y) \in GF(2^m) \times GF(2^m)$$

- Let, $P=(x_1, y_1)$ be a point on the curve.
- $-P=(x_1, x_1 + y_1)$
- Let, $R=P+Q=(x_3, y_3)$

1. Point addition and doubling each require 1 inversion & 2 multiplications

2. We neglect the costs of squaring and addition

Montgomery noticed that the x-coordinate of 2P does not depend on the y-coordinate of P.

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2} \right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, P = Q$$

$$x_1^2 + \frac{b}{x_1^2}, P = Q$$

$$y_3 = \left(\frac{y_1 + y_2}{x_1 + x_2} \right) (x_1 + x_2) + x_1 + y_1, P = Q$$

$$x_1^2 + (x_1 + \frac{y_1}{x_1})x_2 + x_1, P = Q$$

So, now, let us actually study that, I mean, a small point about the scalar multiplication. Now, let us go into the implementations of the elliptic curve point addition and the point doubling operations. So, this is the recapitulation of the equations that we got in the first class. So, this is the addition of P and Q, when P and Q are not the same. So, this is the equation of adding x_1, y_1 and x_2, y_2 and if the points P and Q are same, then, this is the corresponding doubling point. Similarly, the corresponding y coordinates are shown here. Now, the question is, how do we do this operation? So, where do you see that point addition? And another point which is to be noted here, that is, if P is equal to x_1, y_1 , then, minus P is shown as $x_1, x_1 + y_1$.

So, how do you get this minus P? If you take this curve, you take any point x_1, y_1 and draw a vertical line through this x_1, y_1 . So, wherever it intersects the elliptic curve, that is the negative form. So, you can take this and you can solve this, I mean, take a vertical point and say, call it y, I mean, it is a vertical point, I mean, vertical line, and that we have to basically, simultaneously solve with these equations and then you can easily show that, the corresponding minus point is minus P is equal to $x_1, x_1 + y_1$. So, I am not going to the derivation, which you can do yourself; you can take it as an exercise.

So, now, I want to add these two points are, I mean P and Q and store the result in the register R. So, you note that, point addition and doubling, each requires one inversion and two multiplication operation. So, if you observe your addition and doubling here, you see that, you need do this $1 \text{ by } x_1 \text{ plus } x_2$. So, that, that means, that is one inversion operation and that you need to multiply with $y_1 \text{ plus } y_2$; in this case, if I just consider the addition operation; in this case, we need to multiply with, also with $x_1 \text{ plus } x_3$; that is one more multiplication. So, note that multiplications with constants and multiplication and squarings are neglected; because, that I believe that, you can, that we believe that, you, we can also do it without multipliers.

So, the other thing is, if you see that corresponding doubling equation, there also you need to do apply this $1 \text{ by } x_1$; that is, one inversion is necessary. Similarly, is the same $1 \text{ by } x_1$ can be used here also; the other thing is, multiplication with this x_3 . So, that is, the corresponding three, that is you required; you also required to multiply this with this y_1 . So, that is another multiplication. So, which means, you need two multiplications and one inversion, for both point addition and for both point doubling operations. So, we neglect the costs of squaring and addition, and we also neglect the cost of multiplying with constants; like, here you have multiplied with b, that is neglected. So, if b is already a pre known value, or fixed value for that curve, then, you can actually develop an architecture which is devoid of multiplication.

Now, the first thing, which was interesting behind what Montgomery noticed is that, the x coordinate of $2P$ does not depend on the y coordinate of P. So, if you note that, this P equal to Q point, when, I mean, when you are doing a doubling operation, then the x coordinate of these output, actually does not depend upon the y coordinate. So, this was the important observation; that is, you see that, the x coordinate of $2P$, you see that, does not depend upon the corresponding y coordinate; that is it does not depend upon y_1 ; it is only a function of x_1 .

(Refer Slide Time: 15:51)

Montgomery's method to perform scalar multiplication


- Input: $k > 0, P$
- Output: $Q = kP$

1. Set $k \leftarrow (k_{l-1}, \dots, k_1, k_0)_2$
2. Set $P_1 = P, P_2 = 2P$
3. For i from $l-2$ to 0
 - If $k_i = 1$,
Set $P_1 = P_1 + P_2, P_2 = 2P_2$
 - else
Set $P_2 = P_2 + P_1, P_1 = 2P_1$

Return $Q = P_1$

Invariant Property:
 $P = P_2 - P_1$

Question: How to implement the Operation efficiently?



So, based upon this, Montgomery developed, and this was a very important observation, developed a faster method to perform the scalar multiplication. So, this is actually based upon an invariant property, where you say that, there are two points P_2 and P_1 and I want to compute k into P . So, I choose, I rather, I generate P_2 and P_1 , at each step or at each iteration, such that, the difference of P_2 and P_1 is always maintained to be P . So, I want to compute this kP . So, I, first of all, encode this in this fashion and I set P_1 as big P and this P and I set P_2 as $2P$ and so, you see that, P_2 minus P_1 is what? is P . And therefore, what we do is, when we vary from i from l minus 2 to 0 . So, what is this? This is the MSB first approach.

Then, if k_i is equal to 1 , then, what you are doing is that, you are, in P_1 , you are adding; that is, you are doing P_1 plus P_2 ; and in P_2 , you are doubling; that is, you are doing 2 into P_2 . But if this k_i is 0 , then, you are actually adding in the P_2 register, but while you are doubling in the P_1 register. So, you see that, whether k_i is 1 or whether k_i is 0 , P_2 minus P_1 is always invariant. So, you see that, P_2 minus P_1 here is $2P_2$ minus P_1 plus P_2 , which is what? Which is P_2 minus P_1 . Similarly, here P_2 minus P_1 is also P_2 plus P_1 minus $2P_1$, which is again P_2 minus P_1 . Therefore, P_2 minus P_1 is actually an invariant for this algorithm. You see that?


So, now, we are actually doing this addition and doubling in this fashion and therefore, the idea is that, P_1 actually finally, stores the value of k into P . So, what are the

implications of this algorithm? That the, how essentially or what is the relation, I mean, what is the efficiency factor, I mean, how to implement these operations efficiently?

(Refer Slide Time: 17:55)

Example

<p>Compute 7P</p> <ul style="list-style-type: none"> • $7=(111)_2$ • Initialization: <ul style="list-style-type: none"> $P_1=P; P_2=2P$ • Steps: <ul style="list-style-type: none"> – $P_1=3P, P_2=4P$ – $P_1=7P, P_2=8P$ 	<p>Compute 6P</p> <ul style="list-style-type: none"> • $7=(110)_2$ • Initialization: <ul style="list-style-type: none"> $P_1=P; P_2=2P$ • Steps: <ul style="list-style-type: none"> – $P_1=3P, P_2=4P$ – $P_2=7P, P_1=6P$
--	--



So, first of all, let us see that, indeed it computes what we want; that is, the correctness of the algorithm. So, I want to compute 7P. Therefore, first of all, I choose P 1 as P and P 2 as 2P, as we have seen previously. And the steps are, because first we start with this one. Therefore, you see that, this is 1. So, in P 1, what are you doing, you are adding; you are adding P with 2P; you are getting 3P. What about this P 2? P 2, we are doubling. Next time, you are again getting a 1. So, again in P 1, you are adding 3P plus 4 P is 7P and P 2, you are doubling; that is inconsequential in this case. In this case, it is 1 1 0. Therefore, in 1, again you are doing 3P and 4 P like previously, but next time, it is 0. So, you are actually adding in P 2. Therefore, P 2 is 3P plus 4 P which is 7P, where as P 1, while P 1 is double of P 1, that is 6P.

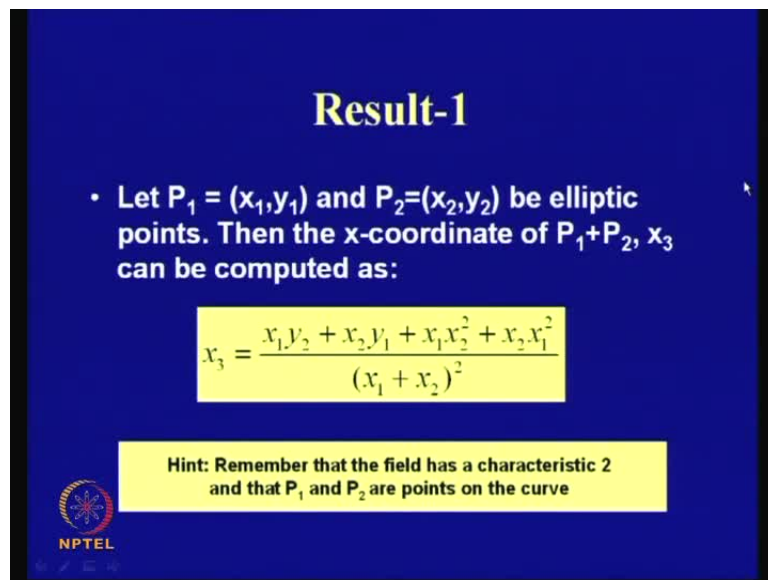
So, again you are returning P 1 as the result. So, you see that, the correctness of the algorithm is still maintained, right. Therefore, indeed P 1 stores the value of k multiplied by P. Now, what is the, I mean, rather, why essentially is this efficient? That is the important question, right. So, rather, why essentially do we do in this fashion?

(Refer Slide Time: 19:10)



So, therefore, that brings us to this point topic of fast multiplication on elliptic curves without pre-computation; that is, I am not having any pre-computed values; then, still I am accelerating the computation of λP or kP into P .

(Refer Slide Time: 19:28)



So, for that, there are certain results which are important, which I will try to discuss here. That is, suppose... So, for that, first of all, let us observe the corresponding sums here.

(Refer Slide Time: 19:41)

$(x_1, y_1), (x_2, y_2)$

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2} \right)^2 + \left(\frac{y_1 + y_2}{x_1 + x_2} \right) + x_1 + x_2 + a$$

$$= \frac{y_1^2 + y_2^2 + (x_1 y_1 + x_1 y_2 + x_2 y_1 + x_2 y_2) + x_1^3 + x_2^3 + x_1^2 x_2 + x_2^2 x_1 + a x_1^2 + a x_2^2}{(x_1 + x_2)^2}$$
 ∵ (x_1, y_1) and (x_2, y_2) are points on the EC:
 $y_1^2 + x_1 y_1 = x_1^3 + a x_1^2 + b$
 $y_2^2 + x_2 y_2 = x_2^3 + a x_2^2 + b$

$$= \frac{(x_1 y_2 + x_2 y_1) + (y_1^2 + x_1 y_1 + x_1^3 + a x_1^2 + b) + (y_2^2 + x_2 y_2 + x_2^3 + a x_2^2 + b)}{(x_1 + x_2)^2} + (x_1^2 + x_2^2 + x_1 x_2)$$

So, the sums of, you know that, x_3 is equal to... So, I am just writing down the results. So, I am adding x_1 comma y_1 and if I add x_2 comma y_2 , the corresponding sum is stored as y_1 plus y_2 divided by x_1 plus x_2 whole squared plus y_1 plus y_2 divided by x_1 plus x_2 , plus x_1 plus x_2 plus a , right; that is, the sums, I mean, that is the corresponding sum, x axis; x coordinate; and the corresponding y coordinate is as shown here as y_1 plus y_2 divided by x_1 plus x_2 into x_1 plus x_2 plus x_3 plus x_3 plus y_1 .

So, at this point, let us just concentrate on this term; that is, the x coordinates of the result. So, I want to add this point x_1 and this point x_2 , y_2 . So, note, at this point, let us make a kind of restriction also is that; let us only be concerned about elliptic curves, which are all points in characteristic 2; that is the elements are chosen from \mathbb{F}_m to the power of m , for any m , for some m .

So, now, if I want to obtain the corresponding sum of this x_1 and y_1 or rather x_1 y_1 and x_2 y_2 , then, I can actually simplify this and I can write it as, x_1 plus x_2 whole squared and the numerator will be as, y_1 squared plus y_2 squared, why, because $2 y_1 y_2$ is, in 0, right here. So, then you have got here, x_1 plus x_2 into y_1 plus y_2 . So, that becomes here $x_1 y_1$ plus $x_1 y_2$ plus $x_2 y_1$ plus $x_2 y_2$ right, plus x_1 plus x_2 whole squared right, rather x_1 plus x_2 whole cubed. So, it is x_1 cubed plus x_2 cubed plus x_1 squared x_2 plus x_2 squared x_1 , right, x_2 squared x_1 , this is correct.

So, x_1 cubed plus, we are multiplying this with this, right. So, x_1 cubed plus x_2 cubed plus x_1 squared x_2 plus x_2 squared x_1 plus a multiplied with x_1 squared plus a multiplied with x_2 squared, right. And you note that, x_1, y_1 is a point on the elliptic curve. So, you choose, you know that, if this is your equation, then, you know that, since x_1, y_1 and x_2, y_2 are points on the elliptic curve, you know that, y_1 squared plus $x_1 y_1$ is equal to x_1 cubed plus $a x_1$ squared plus b .

Similarly, your y_2 squared plus $x_2 y_2$ is equal to x_2 cubed plus $a x_2$ squared plus b , right. So, that means, if you observe now, these term, then, your numerator becomes, we can actually combine, and you will see that, you have got like $x_1 y_2$ plus $x_2 y_1$, that is this term and this term, plus you can actually write, y_1 squared plus $x_1 y_1$ plus x_1 cubed plus $a x_1$ squared plus b plus, I am adding a b there, and y_2 cubed y_2 squared plus $x_2 y_2$ plus x_2 cubed plus $a x_2$ squared plus b . So, if you do this, then, you note that, there are some more terms. So, there, this is that, x_1 squared x_2 plus x_2 squared x_1 . So, this full thing is there in the numerator, right.

(Refer Slide Time: 24:49)

$$\frac{y_1^2 + y_2^2 + (x_1 y_1 + x_1 y_2 + x_2 y_1 + y_1 y_2) + x_1^3 + x_2^3 + a x_1^2 + a x_2^2}{(x_1 + x_2)^2}$$

$\because (x_1, y_1)$ and (x_2, y_2) are points on the EC:

$$y_1^2 + x_1 y_1 = x_1^3 + a x_1^2 + b$$

$$y_2^2 + x_2 y_2 = x_2^3 + a x_2^2 + b$$

$$= \frac{(x_1 y_2 + x_2 y_1) + (y_1^2 + x_1 y_1 + x_1^3 + a x_1^2 + b) + (y_2^2 + x_2 y_2 + x_2^3 + a x_2^2 + b) + (x_1^2 x_2 + x_2^2 x_1)}{(x_1 + x_2)^2}$$

$$= \frac{(x_1 y_2 + x_2 y_1) + (x_1^2 x_2 + x_2^2 x_1)}{(x_1 + x_2)^2}$$

Is it so? That means, this term and this term will go to 0, because they are points on the curve and being in characteristic two field, these two terms go to 0. So, you have got x_1 plus x_2 whole squared in the denominator and in the numerator you have got $x_1 y_2$ plus $x_2 y_1$ plus x_1 squared x_2 plus x_2 squared x_1 , is it correct. So, that is your corresponding sum of x_1, y_1 and x_2, y_2 , right. So, that is the first result here,

which says that, here, if you take x_1, y_1 and if you take x_2, y_2 , and these are points on elliptic curve, then x coordinate of $P_1 + P_2, x_3$ can be computed as $x_1 y_2 + x_2 y_1 + x_1 x_2^2 + x_2 x_1^2$ divided by $x_1 + x_2$ whole squared.


(Refer Slide Time: 25:37)

Result-1

- Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be elliptic points. Then the x -coordinate of $P_1 + P_2, x_3$ can be computed as:

$$x_3 = \frac{x_1 y_2 + x_2 y_1 + x_1 x_2^2 + x_2 x_1^2}{(x_1 + x_2)^2}$$

Hint: Remember that the field has a characteristic 2 and that P_1 and P_2 are points on the curve



So, therefore, you remember that the field has got a characteristic two and that P_1 and P_2 are points on the curve. So, these are the two things that we have used in this result.

(Refer Slide Time: 26:12)

© CET
I.I.T. KGP

$P = P_2 - P_1$ — invariant.


$(x, y) = (x_2, y_2) + (x_1, x_1 + y_1)$

$$x = \frac{x_1 y_2 + x_2 (x_1 + y_1) + (x_1 x_2^2 + x_2 x_1^2)}{(x_1 + x_2)^2}$$

$$x_3 = \frac{x_1 y_2 + x_2 y_1 + x_1 x_2^2 + x_2 x_1^2}{(x_1 + x_2)^2}$$

$$x + x_3 = \frac{x_1 x_2}{(x_1 + x_2)^2}$$

$$x_3 = x + \frac{x_1 x_2}{(x_1 + x_2)^2} = x + \frac{x_1^2}{(x_1 + x_2)^2} + \frac{x_2^2}{(x_1 + x_2)^2}$$



So, therefore, that from this, that if I, if in this Montgomery's algorithm, that we know that P was maintained to be equal to P^2 minus P^{-1} ; that was an invariant for the Montgomery's algorithm, right. Therefore, P^2 was essentially a point like x^2 and y^2 and what was P^{-1} ? P^{-1} was equal to x^{-1} comma x^{-1} plus y^{-1} , I mean, minus of P^{-1} is equal to x^{-1} comma x^{-1} plus y^{-1} . So, therefore, the sum of these two, that are of x^2 y^2 and x^{-1} comma x^{-1} plus y^{-1} is nothing,, but x comma y . So, this is the sum of these two points, right. So, therefore, we know that from here, your x , if you say that, this x is equal to, I mean, you know that, this x , you can actually write this x as x^{-1} plus x^2 whole squared into x^{-1} y^2 plus x^2 . So, you have, you can actually use this y co-ordinate as x^{-1} plus y^{-1} plus x^{-1} x^2 squared plus x^2 x^{-1} squared, right. Basically, I am adding up these two things and using the previous result.

So, similarly, you can add up these terms. So, and if you want the sum of P^{-1} and P^2 ... So, you call it P^{-1} plus P^2 and call it as x^3 comma y^3 , then your x^3 plus what we have previously got... So, I am again writing that, x^{-1} plus x^2 whole squared is x^{-1} y^2 plus x^2 y^{-1} plus x^{-1} squared x^2 plus x^2 squared x^{-1} ; this was what, x^3 . So, you can add these two equations, where you will get like, x plus x^3 is equal to x^{-1} plus x^2 whole squared and in the numerator, you will see that, x^{-1} y^2 x^2 y^{-1} x^{-1} squared x^2 and x^2 squared x^{-1} all of them will cancel. So, what you will be remaining with, is only x^{-1} x^2 .

So, therefore, you can actually write x^3 , as you can rearrange these terms and you can express x^3 as x plus x^{-1} plus x^2 whole squared into x^{-1} x^2 . Now, you see that, if you want to do this operation, then, how many you need to multiply this, right. Therefore, the other way of writing this, in an equivalent fashion like, x plus x^{-1} plus x^2 whole squared x^{-1} squared plus x^2 divided by x^{-1} plus x^2 .

So, this is simple. You see that, if I can take x^{-1} plus x^2 whole squared, then x^{-1} squared will stay, x^2 will be multiplied with x^{-1} plus x^2 , so, x^2 squared will be there and the other term will be x^{-1} into x^2 , right. So, you see this, sorry, this is 1, this is x^{-1} , right.

So, if in this case, this x^{-1} squared and this x^{-1} squared will be get cancelled, right. Now, what is the advantage of doing this in this fashion? So, how many inversions do you need to do? You need to do one inversion, because you need compute 1 by x^{-1} plus x^2 . And, how many multiplications you need to do? You need to multiply with only x^{-1} . You need

to, I want to compute the x^{-1} , sorry... I, basically, do a 1 by x^{-1} plus x^{-2} ; that is 1 inversion; I multiply it with x^{-1} . That is how many multiplications? One multiplication.

And then, I need to do a squaring. So, how many multiplications I did? One multiplication. But here, I have to multiply with x^{-2} , I have to multiply with x^{-1} . So, I needed two multiplication operation. So, you see that, it is quite interesting; the same thing, but if you just rewrite and expand little bit, you see that, you are basically saving the number of multiplication operations that you need to do.


(Refer Slide Time: 31:01)

Result-2

- Let $P=(x,y)$, $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ be elliptic points. Let $P=P_2-P_1$ be an invariant.

Then the x-coordinate of P_1+P_2 , x_3 can be computed in terms of the x-coordinates as:

$$x_3 = \left\{ \begin{array}{l} x + \left(\frac{x_1}{x_1 + x_2} \right)^2 + \frac{x_1}{x_1 + x_2}, P_1 = P_2 \\ x_1^2 + \frac{b}{x_1^3}, P_1 = P_2 \end{array} \right.$$



And therefore, that is your result 2; that is, if your P is equal to P_2 minus P_1 , then the x co-ordinate of P_1 plus P_2 , x_3 can be computed in terms of the x co-ordinate as this. So, when you see that, here, whenever you are computing this x_3 , that is, whenever you are computing P_1 plus P_2 , you are not actually bothered about the y co-ordinate; you see that, right; that means, you can do the operation entirely devoid of y co-ordinate.

So, previously, we had already noticed, Montgomery noticed, that the doubling was actually devoid of the y co-ordinate; but then, you, it was also a used, I mean, the thing which is used here is that, if you maintain this invariant, that is, if P is equal to P_2 minus P_1 , then the sum of P_1 plus P_2 is also made devoid of the y co-ordinate; that means, you are not actually operating on 2×2 co-ordinates; we are actually operating on only


the x co-ordinate. So, that means, you need not care about the y co-ordinate, if you are maintaining this. And that, actually gives you an amount of efficiency.

(Refer Slide Time: 32:02)

Result-3

Let $P=(x,y)$, $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ be elliptic points. Assume that $P_2-P_1=P$ and x is not 0. Then the y-coordinates of P_1 can be expressed in terms of P , and the x-coordinates of P_1 and P_2 as follows:

$$y_1 = (x_1 + x) \{ (x_1 + x)(x_2 + x) + x^2 + y^2 \} / x + y$$



Then, you also need to obtain a y co-ordinate finally, right. So, for that, this result is used, that is, if P is equal to x comma y , and your P_1 is x_1 comma y_1 and P_2 is x_2 comma y_2 , the elliptic points and assume that, P_2 minus P_1 is again equal to P , and x is not 0, then the y co-ordinate of P_1 can be expressed in terms of P and the x co-ordinates of P_1 and P_2 as follows.

(Refer Slide Time: 32:39)

© CET
I.I.T. KGP


$$P = P_2 - P_1 \Rightarrow P_2 = P + P_1$$

(x_2, y_2)
/

(x, y)
+

(x_1, y_1)
/

$$x_2 = \frac{x_1 y + x y_1 + x_1 x^2 + x x_1^2}{(x_1 + x)^2}$$

$$\begin{aligned} x_1 y_1 &= x_2 (x_1 + x)^2 + x_1 y + x_1 x^2 + x x_1^2 \\ &= x_2 (x_1^2 + x^2) + x_1 y + x_1 x^2 + x x_1^2 \\ &= x_1 \{ x_1 x_2 + x_1 x + x^2 + y \} + x \{ x x_2 \} \\ &= x_1 \left\{ \frac{x_1 x_2 + x_1 x + x^2 + y}{x_1 + x} + \frac{x x_2 + x^2 + y}{x_1 + x} \right\} \\ &= (x_1 + x) \{ (x_1 + x)(x_2 + x) + x^2 + y^2 \} + x y \end{aligned}$$



So, that you can write as, if you take x_1 plus x common... So, you see that, here x squared and x squared will get canceled out; and, there is a plus y term here. So, there is an additional plus y term here. So, if you see that, this one is x_1 into x_2 plus x_1 into x plus x_2 plus y . What is this term here? x_2 plus x_2 x_1 plus x_1 plus y . That is the same term, basically. So, therefore, you can take this as common and you can write this as, x_1 plus x into x_2 plus x plus x squared plus y plus x into y .

(Refer Slide Time: 38:06)

Result-3

Let $P=(x,y)$, $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ be elliptic points. Assume that $P_2 - P_1 = P$ and x is not 0. Then the y -coordinates of P_1 can be expressed in terms of P , and the x -coordinates of P_1 and P_2 as follows:

$$y_1 = (x_1 + x) \{ (x_1 + x)(x_2 + x) + x^2 + y \} / (x + y)$$



NPTEL

So, therefore, now you can actually obtain y_1 from here. You can just take and divide this by x , you should get y_1 . That is, what is, you remember here, that is, y_1 is equal to x_1 plus x multiplied with x_1 plus x into x_2 plus x plus x squared plus y divided by x plus x y by x . So, that is y_1 . So, that means, that you are always doing, wherever in this Montgomery's algorithm you are only bothered about the x co-ordinates and you are doing it; finally, at the end when you need the y co-ordinate of the output also, you can apply this equation to get the y co-ordinate.


(Refer Slide Time: 38:34)

Final Algorithm

Input: $k > 0$, $P = (x, y)$
Output: $Q = kP$

1. If $k = 0$ or $x = 0$ then output $(0, 0)$
2. Set $k = (k_{l-1}, k_{l-2}, \dots, k_0)_2$
3. Set $x_1 = x$, $x_2 = x^2 + b/x^2$ • #INV: $2(l-2)+1$;
4. For i from $l-2$ to 0 • #MULT: $2(l-2)+4$
 1. Set $t = x_1/(x_1 + x_2)$ • #ADD: $4(l-2)+6$
 2. If $k_i = 1$, • #SQR: $2(l-2)+2$

$$x_1 = x + t^2 + t, \quad x_2 = x_2^2 + b/x_2^2$$
 - else
$$x_1 = x_1^2 + b/x_1^2, \quad x_2 = x + t^2 + t$$
5. $r_1 = x_1 + x$, $r_2 = x_2 + x$
 $y_1 = r_1(r_1 r_2 + x^2 + y)/x + y$
Return $Q = (x_1, y_1)$



So, therefore, if you write all these things in the form of an algorithm, then, this is the way, how you can do so. So, you see that, here we start with x_1 equal to x , that is the x co-ordinate of the point. And what does this indicate, x_2 equal to $x^2 + b/x^2$? What does this indicate? This indicates it is $2P$. And I am only bothered about the x co-ordinate. I am not bothered about the y co-ordinate. So, x_2 is equal to $x^2 + b/x^2$ is only the x co-ordinate of twice P . You remember, the algorithm, this is the Montgomery algorithm.

(Refer Slide Time: 39:07)

Montgomery's method to perform scalar multiplication


- Input: $k > 0$, P
- Output: $Q = kP$

1. Set $k = (k_{l-1}, \dots, k_1, k_0)_2$
2. Set $P_1 = P$, $P_2 = 2P$
3. For i from $l-2$ to 0
 - If $k_i = 1$,
$$\text{Set } P_1 = P_1 + P_2, P_2 = 2P_2$$
 - else
$$\text{Set } P_2 = P_2 + P_1, P_1 = 2P_1$$

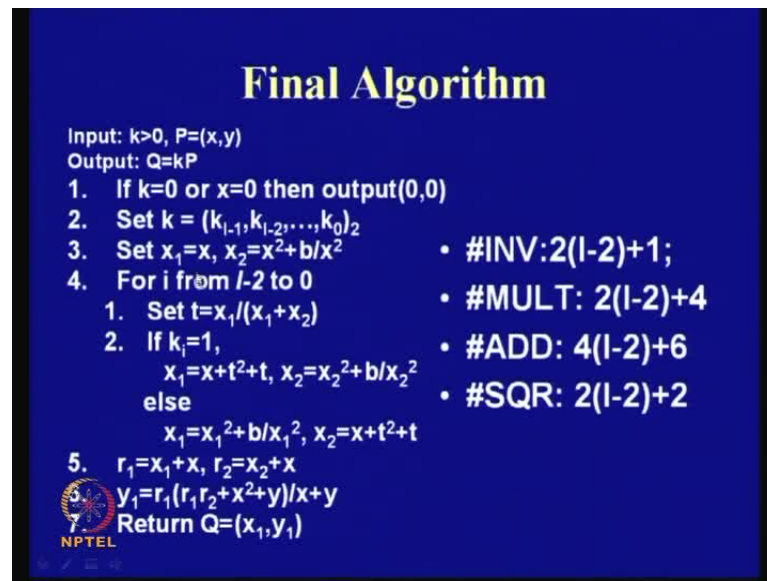
Return $Q = P_1$

Invariant Property:
 $P = P_2 - P_1$

Question: How to implement the Operation efficiently?



(Refer Slide Time: 39:19)



Final Algorithm

Input: $k > 0$, $P = (x, y)$
Output: $Q = kP$

1. If $k=0$ or $x=0$ then output $(0,0)$
2. Set $k = (k_{l-1}, k_{l-2}, \dots, k_0)_2$
3. Set $x_1 = x$, $x_2 = x^2 + b/x^2$ • #INV: $2(l-2)+1$;
4. For i from $l-2$ to 0 • #MULT: $2(l-2)+4$
 1. Set $t = x_1/(x_1 + x_2)$ • #ADD: $4(l-2)+6$
 2. If $k_i = 1$, • #SQR: $2(l-2)+2$
 $x_1 = x + t^2 + t$, $x_2 = x_2^2 + b/x_2^2$
else
 $x_1 = x_1^2 + b/x_1^2$, $x_2 = x + t^2 + t$
5. $r_1 = x_1 + x$, $r_2 = x_2 + x$
 $y_1 = r_1(r_1 r_2 + x^2 + y)/x + y$
Return $Q = (x_1, y_1)$

NPTEL


So, therefore, we had P 1 storing P, and P 2 storing twice P, right. And in this operation, I am only bothered about the x co-ordinates. Therefore, here we only store the, we store the corresponding x co-ordinates; it is x and this is x squared plus b by x squared. Now, what we are doing is that, we are calculating the value of t equal to x 1 by x 1 plus x 2. And then, if this value is 1, then we are doing an addition operation here. The addition is defined only by doing x plus t squared plus t. So, that is only the x co-ordinate of the corresponding sum of P 1 and P 2 of P 1 and P 2 or rather an x 1 and x 2. And here, you are storing the doubling thing and when this value is 0, here, you are doing the doubling in x 1 and you are doing the addition operation in x 2. And, do you understand why it is x plus t squared plus t?

(Refer Slide Time: 40:03)

Result-2

- Let $P=(x,y)$, $P_1 = (x_1,y_1)$ and $P_2=(x_2,y_2)$ be elliptic points. Let $P=P_2-P_1$ be an invariant.

Then the x-coordinate of P_1+P_2 , x_3 can be computed in terms of the x-coordinates as:

$$x_3 = \begin{cases} x + \left(\frac{x_1}{x_1+x_2} \right)^2 + \frac{x_1}{x_1+x_2} \cdot P_1 = P_2 & \\ x_1^2 + \frac{b}{x_1} \cdot P_1 = P_2 & \end{cases}$$



That is, from the result 2, right, that is from the result 2, where you are doing this operation. So, this is your t. So, this is t squared plus t.

(Refer Slide Time: 40:12)

Final Algorithm

Input: $k>0$, $P=(x,y)$
Output: $Q=kP$

- If $k=0$ or $x=0$ then output $(0,0)$
- Set $k = (k_{l-1}, k_{l-2}, \dots, k_0)_2$
- Set $x_1=x$, $x_2=x^2+b/x^2$ • #INV: $2(l-2)+1$;
- For i from $l-2$ to 0
 - Set $t=x_1/(x_1+x_2)$ • #MULT: $2(l-2)+4$
 - If $k_i=1$,
 - $x_1=x+t^2+t$, $x_2=x_2^2+b/x_2^2$ • #ADD: $4(l-2)+6$
 - else
 - $x_1=x_1^2+b/x_1^2$, $x_2=x+t^2+t$ • #SQR: $2(l-2)+2$
 - $r_1=x_1+x$, $r_2=x_2+x$
 $y_1=r_1(r_1+r_2+x^2+y)/x+y$
 Return $Q=(x_1,y_1)$

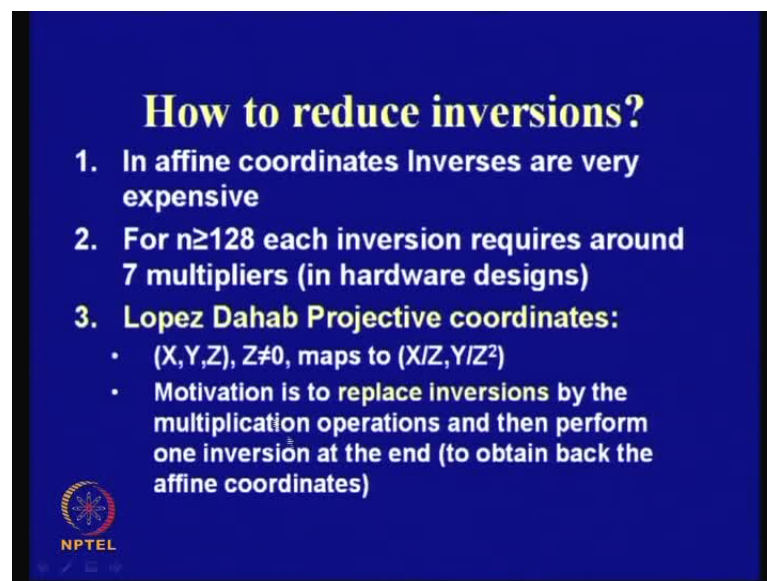


So, you are doing that t squared plus t only to do the addition operation. So, finally, you need to obtain the y co-ordinate. Therefore, you can obtain the y co-ordinate by doing this operation. So, what is the advantage here, you can see, I mean, what is the number of inversions, multiplications, additions and squaring required here, is shown here, which you can work out. Therefore, it is like, if you want to do the inversion, you see that, you

are always doing an inversion operation. So, it is $1 \times 1 + x^2$ means, that is, cost of one inversion always. And whatever you do, you are always doing inversion either here or you are doing a inversion here. So, that means, it is a two inversions which are necessary.

So, there is one here and there is either this one or this one. So, that is two inversions, multiplied by the number of times you are doing this, running this loop, plus 1 because you are doing an another additional inversion operation here. Similarly, you can also count the number of multiplication operations here, additions and squarings. So, you see that, here, still you have got a large number of times you have to do this inversion operation. So, that means, that an, that inversion is actually a very costly step in finite field operations.

(Refer Slide Time: 41:22)



How to reduce inversions?

1. In affine coordinates Inverses are very expensive
2. For $n \geq 128$ each inversion requires around 7 multipliers (in hardware designs)
3. Lopez Dahab Projective coordinates:
 - (X, Y, Z) , $Z \neq 0$, maps to $(X/Z, Y/Z^2)$
 - Motivation is to replace inversions by the multiplication operations and then perform one inversion at the end (to obtain back the affine coordinates)

NPTEL

So, therefore, there was, in order to reduce this number of inversions, these are fine co-ordinate systems were actually converted into something which is called as projective co-ordinates. So, that is basically, a 3 dimensional space which is being borrowed to reduce the number of inversions which are necessary. So, for example, where n is greater than equal to 128, each inversion is actually like equivalent to 7 multipliers in hardware design; that means, it is a, there is a lot of cost and therefore, if there is an importance of reducing the number of inversions. Therefore, there was one projective co-ordinates which is called as the Lopez Dahab projective co-ordinates, where x , y and z are the

three projective systems and x , the affine system small x is actually equal to x by z and a small y is written as y by z squared. So, that is being written as these x comma y comma z equivalence class.

So, the motivation is to, now replace this inversions by the multiplication operations and then, perform one inversion at the end to obtain back the affine coordinates. So, what is done in projective coordinates is that, all the operations, the addition and doubling are done devoid of any inversions. They are done at the cost of increased number of addition operations. And increased number of multiplication operations and finally, there is a necessity of converting the output from the projective coordinates back to the affine coordinates. There we will need to do one more inversion state.

(Refer Slide Time: 42:47)

Doubling


- Remember:

$$x_3 = \left\{ \begin{array}{l} x + \left(\frac{y_1}{x_1 + x_2} \right)^2 + \frac{y_1}{x_1 + x_2} \cdot P_1 = P_3 \\ x_1^2 + \frac{b}{x_1} \cdot P_1 = P_2 \end{array} \right.$$

- 2 inverses
 - 1 general field multiplication
 - 4 additions
 - 2 squarings
- In Projective Coordinates:

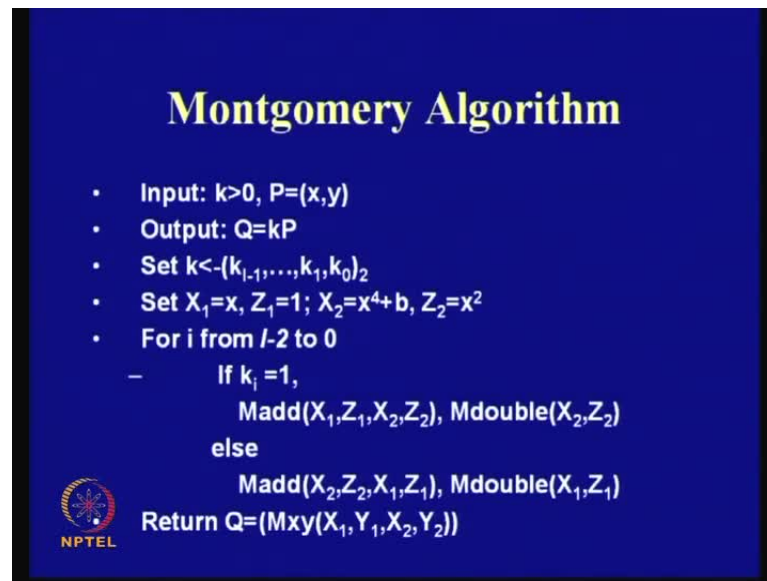
$$\begin{array}{l} P_1 = P_2, X_3 = X_1^4 + b Z_1^4 \\ Z_3 = Z_1^2 \cdot X_1^2 \\ P_1 = P_2, Z_3 = (X_1 Z_2 + X_2 Z_1)^2 \\ X_3 = x Z_3 + (X_1 Z_2)(X_2 Z_1) \end{array}$$

- 0 inverses
 - 4 general field multiplications
 - 3 additions
 - 5 squarings




So, for example, here, if you remember that, in doubling, this was the corresponding equations in Montgomery system. That, these are the values which are being shown here. That, this you need to do 2 inverses, 1 general field multiplication, 4 addition and squaring. If you convert this into the projective coordinates by the previous transformations that we have seen, then, you will see that, you actually need to do 0 inverses. There are no inverses necessary, but you need to do increased number of multiplication and addition and squaring operations. So, therefore, depending upon your n is to i ratio, that is the number of multipliers in your inverses, this may become handy and may help to make your implementations more efficient.

(Refer Slide Time: 43:30)

A blue slide with white text titled "Montgomery Algorithm". It lists the input, output, and steps of the algorithm. The steps include setting k to a binary vector, initializing $X_1=x$, $Z_1=1$, $X_2=x^4+b$, and $Z_2=x^2$. A loop from $i=l-2$ to 0 follows. Inside the loop, if $k_i=1$, it performs $\text{Madd}(X_1, Z_1, X_2, Z_2)$ and $\text{Mdouble}(X_2, Z_2)$; otherwise, it performs $\text{Madd}(X_2, Z_2, X_1, Z_1)$ and $\text{Mdouble}(X_1, Z_1)$. Finally, it returns $Q = (\text{Mxy}(X_1, Y_1, X_2, Y_2))$. An NPTEL logo is in the bottom left corner.

Montgomery Algorithm

- Input: $k > 0$, $P = (x, y)$
- Output: $Q = kP$
- Set $k \leftarrow (k_{l-1}, \dots, k_1, k_0)_2$
- Set $X_1 = x$, $Z_1 = 1$; $X_2 = x^4 + b$, $Z_2 = x^2$
- For i from $l-2$ to 0
 - If $k_i = 1$,
 $\text{Madd}(X_1, Z_1, X_2, Z_2)$, $\text{Mdouble}(X_2, Z_2)$
 - else
 $\text{Madd}(X_2, Z_2, X_1, Z_1)$, $\text{Mdouble}(X_1, Z_1)$
- Return $Q = (\text{Mxy}(X_1, Y_1, X_2, Y_2))$



So, therefore, if you convert this entire system into projective coordinates, this is how the Montgomery's algorithm looks like. So, here you see that, here, you have set here x as a , I mean x , I mean X_1 as small x , Z_1 as 1 and X_2 is x to the power of 4 plus b and Z_2 is x squared. So, x to the power of 4 plus b is... So, you see that, X_2 and Z_2 are storing what? Are storing the double point and here it is storing the single point. Now, you are doing, if k_i is equal to 1 , you are doing in projective coordinates, the addition operations in $X_1 Y_1$ and you are a doing doubling in $X_2 Z_2$.

So, you see that, you are not actually working on the y_2 point. Why? Because, your Montgomery's algorithm, Montgomery's technique does not need me to work on the y coordinate; it just needs only the x coordinate. And when your k_i is 0 , then, also you need to do the addition here and the doubling in the $X_1 Z_1$ register. And finally, when you have got the result, then you need to convert this back into the affine coordinates.


(Refer Slide Time: 44:38)

Mxy: Projective to Affine

$$x_3 = X_1 / Z_1$$

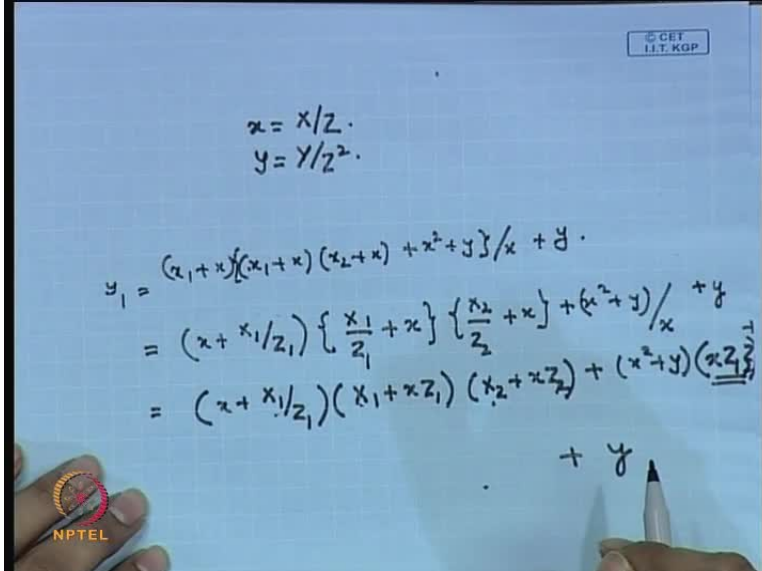
$$y_3 = (x + X_1 / Z_1) [(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)] (xZ_1Z_2)^{-1} + y$$

Requires 10 multiplications and one inverse operation



So, for that, actually I mean, I am not going to this, but you need to do one, one inversion at that point, is necessary. So, this is actually straight forward, because this is exactly like the previous algorithm. You had basically this step.

(Refer Slide Time: 44:58)




© CET
I.I.T. KGP

$$x = x/z.$$

$$y = y/z^2.$$

$$y_1 = \frac{(x_1 + x)(x_1 + x)(x_2 + x) + x^2 + y}{x + y}$$

$$= (x + x_1/z_1) \left\{ \frac{x_1}{z_1} + x \right\} \left\{ \frac{x_2}{z_2} + x \right\} + (x^2 + y) / x + y$$

$$= (x + x_1/z_1)(x_1 + xz_1)(x_2 + xz_2) + (x^2 + y)(xZ_1Z_2)^{-1} + y$$


So, see for example, you had y_1 as x_1 plus x into... So, basically, what you did there was, if you see the y_1 equation, was x_1 plus x into x_1 plus x , into x_2 plus x plus x squared plus y , this was divided by x plus y ; that was your y coordinates. So, now, your transformation in your projective coordinate system is x is equal to x by z ; your y is y by

z squared. So, if you do this, then, what you will write here is, you will take this point; therefore, you see that, if you want to do this conversion, you will write here this x as x, this x 1 point, you will right as x 1 by z 1. And your x 1 plus x, again you will write as x 1 by z 1 plus x. x 2 you will write as, x 2 by z 2 plus x, right. plus x squared plus y, right. And that is divided by x and again point y is added, I mean y is added.

So, therefore, this you can actually simplify and you can write as x plus x 1 by z 1 and this, as x 1 plus x into z 1 x 2 plus x into z 2. So, you see that, there is a z 1 by z 2 at the bottom. Then, you have got x squared plus y and you can actually write all these z 1 z 2 z 1 z 2 and minus, inverse right, therefore, x into z 1 z 2 plus this term y. So, therefore, you see that, all these terms x 1 and your x 2 and your x 1 and your x 2 and your z 1, z 2 are there in the projective coordinates. You are using this projective coordinates and you are basically applying only one inversion operation to finally, convert your projective result back into the affine coordinates. And, the corresponding x thing is very simple, because you just need to divide this by the corresponding z to obtain the x coordinate.

(Refer Slide Time: 44:38)

Mxy: Projective to Affine

$$x_3 = X_1 / Z_1$$

$$y_3 = (x + X_1 / Z_1) [(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)] (xZ_1Z_2)^{-1} + y$$

Requires 10 multiplications and one inverse operation

NPTEL


So, therefore, the if you want to obtain x 3, that is a result of P 1 and P 1 plus P 2, the x coordinate, you just need to divide x 1, projective coordinate x 1 by z 1, and for the y coordinate, you need to do more operations. Actually, you need to do 10 multiplications and 1 inversion operation to do the final transformation.

(Refer Slide Time: 48:07)

Final Comparison

<u>Affine Coordinates</u>	<u>Projective Coordinates</u>
Inv: $2\log k + 1$	Inv: 1
Mult: $2\log k + 4$	Mult: $6\log k + 10$
Add: $4\log k + 6$	Add: $3\log k + 7$
Sqr: $2\log k + 2$	Sqr: $5\log k + 3$

Hence, final decision depends upon the I:M ratio of the finite field operators



So, if you do a neck to neck comparison of affine coordinates and projective coordinates, this is how its looks like. There is more, the inversions have been reduced, but there are multiplications and the additions and the squarings have of course, increased. Therefore, whether you will go for your affine coordinates or whether you will go for your projective coordinates depends upon your i is to m ratio; that is, in your inverse how many multiplications are multipliers are there. So, that actually dictates, whether this one is more efficient or whether this one is more efficient.

(Refer Slide Time: 48:40)

Addition in Mixed Coordinates


- Theorem: Let $P_1=(X_1/Z_1, Y_1/Z_1^2)$ and $P_2=(X_2/Z_2, Y_2/Z_2^2)$ be two points on the curve. If $Z_1=1$, then $P_1+P_2=(X_3/Z_3, Y_3/Z_3^2)$ st.

$$U = Z_2^2 Y_1 + Y_2, S = Z_2 X_1 + X_2, T = Z_2 S, Z_3 = T^2,$$

$$V = Z_2 X_1, X_3 = U^2 + T(U + S^2 + T a),$$

$$Y_3 = (V + X_3)(TU + Z_2) + Z_2^2 C$$

Number of multiplications are further reduced. Squaring is increased a bit, but they are cheap in $GF(2^n)$ Improvement by 10 % if $a \neq 0$, otherwise 12 %...



So, there are, I mean, I will not go in to this, that is, there are some approaches which says that, you actually keep one of the points in projective coordinates and keep the other point in affine coordinates; that is called as mixed coordinate systems. So, there are, these are some equations we have derived, which you can check also offline, that is, but in this case, the main thing we pointed out here is that, the number of multiplications are further reduced, squaring is increased a bit, but they are cheap in GF 2 power of n. There is 10 percent improvement if a is not equal to 0 and if, so, a is 0, then, there is a 12 percent improvement. So, mixed coordinate systems are at times more efficient than even the projective coordinates systems.

(Refer Slide Time: 49:22)

Parallel Strategies for Scalar Point Multiplication

- **Point Doubling**
 - Cycle 1: $T=X_1^2, M=cZ_1^2, Z_2=T.Z_1^2$
 - Cycle 1a: $X_2=T^2+M^2$ 1 multiplier
- **Point Addition**
 - Cycle 1: $t_1=(X_1.Z_2); t_2=(Z_1.X_2)$
 - Cycle 1a: $M=(t_1+t_2), Z_1=M^2$ 2 multipliers
 - Cycle 2: $N=t_1.t_2, M=xZ_1$
 - Cycle 2a: $X_1=M+N$

NPTEL We assume that squarings and multiplications with constants can be performed without multipliers...

So, the, I will just finally, conclude with some parallelizing, comments on parallelizing strategies on the point doubling and the point addition operation. So, you see that, in your point doubling... So, I am considering the projective coordinates here. So, in, these are the operations that you did. Therefore, you just think of x 1 comma y 1, what you have done here, is this. That is, you have taken this x 1, squared it, see z 1 squared z 2 is t into z 1 squared and finally, you are adding this t squared plus m squared and this is your doubled point. This you can check, that is, this is exactly the same as what we did previously.

So, how many multiplications are necessary here? There is one multiplication operation necessary. Multiplication is constant, I am not again considering as multiplication. So, if


I got one multiplier, then, that is sufficient to do this operation, but what about doubling? You see that, there are more number of steps. So, here for example, there is one multiplication, here is another multiplication. So, if I want to do a parallelization, then I have two, and there is again a multiplication here, again a multiplication here. So, if I got 2 multipliers, then, I can actually parallelize this step and I can parallelize these two steps also, right. Therefore, depending upon my resource constant, I can actually parallelize this point addition also.

(Refer Slide Time: 50:46)

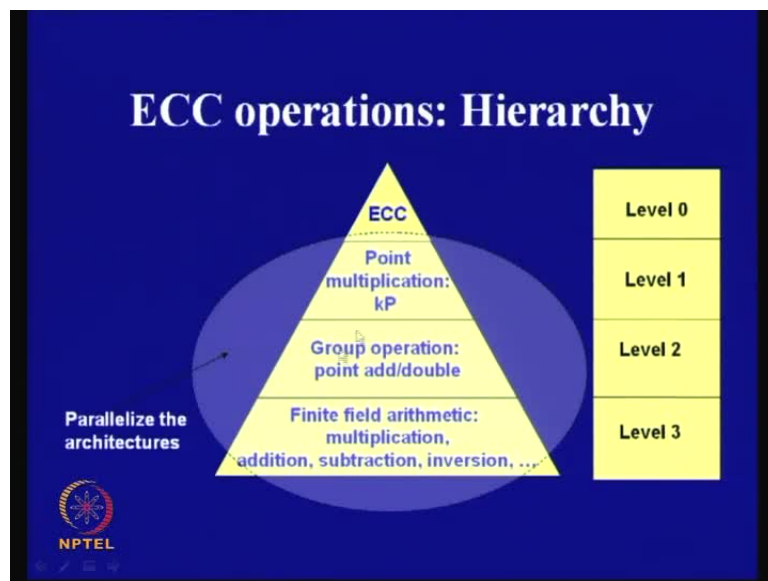
Parallelizing Montgomery Algorithm

1. Input: $k > 0, P = (x, y)$
2. Output: $Q = kP$
3. Set $k < (k_{l-1}, \dots, k_1, k_0)_2$
4. Set $X_1 = x, Z_1 = 1; X_2 = x^4 + b, Z_2 = x^2$
5. For i from $l-2$ to 0
 - If $k_i = 1,$
 - 5a) $\text{Madd}(X_1, Z_1, X_2, Z_2), \text{Mdouble}(X_2, Z_2)$
 - else
 - 5b) $\text{Madd}(X_2, Z_2, X_1, Z_1), \text{Mdouble}(X_1, Z_1)$

Return $Q = (\text{Mxy}(X_1, Y_1, X_2, Y_2))$

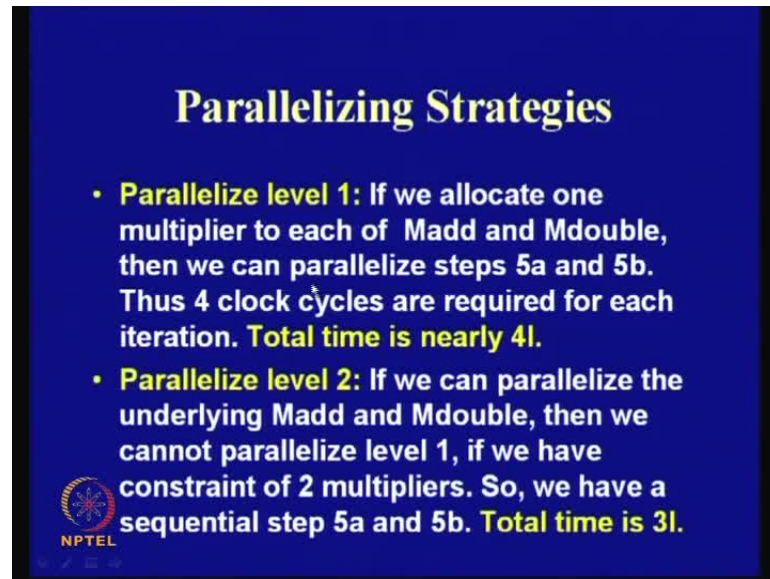


(Refer Slide Time: 02:21)



And what will be the final, I mean, if you just see of this Montgomery's algorithm structure, if I want to parallelize this, I can parallelize this probably at various levels. Like I can either parallelize this, at this level of kP or I can the parallelize the inherent internal doubling and addition operations; that is, I can either parallelize level 1 or I can parallelize level 2.

(Refer Slide Time: 51:10)



Parallelizing Strategies


- **Parallelize level 1:** If we allocate one multiplier to each of Madd and Mdouble, then we can parallelize steps 5a and 5b. Thus 4 clock cycles are required for each iteration. **Total time is nearly 4l.**
- **Parallelize level 2:** If we can parallelize the underlying Madd and Mdouble, then we cannot parallelize level 1, if we have constraint of 2 multipliers. So, we have a sequential step 5a and 5b. **Total time is 3l.**

So, here are some comments. Like, suppose, if we allocate 1 multiplier to each of m add and m double, one multiplier to each of m add and m double, then, we can parallelize steps 5 a and 5 b. So, we have got two multipliers; we are allocating one to the addition and one to the doubling.

(Refer Slide Time: 50:46)

Parallelizing Montgomery Algorithm

1. Input: $k > 0, P = (x, y)$
2. Output: $Q = kP$
3. Set $k < \{k_{l-1}, \dots, k_1, k_0\}_2$
4. Set $X_1 = x, Z_1 = 1; X_2 = x^4 + b, Z_2 = x^2$
5. For i from $l-2$ to 0
 - If $k_i = 1,$
 - 5a) $\text{Madd}(X_1, Z_1, X_2, Z_2), \text{Mdouble}(X_2, Z_2)$
 - else
 - 5b) $\text{Madd}(X_2, Z_2, X_1, Z_1), \text{Mdouble}(X_1, Z_1)$
6. Return $Q = (\text{Mxy}(X_1, Y_1, X_2, Y_2))$




So, you see that, here, you are actually doing a addition and then, you are a doubling in parallel. Because you have two multipliers, we have actually given one multiplier to this and one multiplier to this and you can actually doing both of them in concurrence, right. So, what will be the cycle length required, will be that, equal to that, that required for the addition because addition is more.

(Refer Slide Time: 49:22)

Parallel Strategies for Scalar Point Multiplication

- Point Doubling
 - Cycle 1: $T = X_1^2, M = cZ_1^2, Z_2 = T \cdot Z_1^2$
 - Cycle 1a: $X_2 = T^2 + M^2$
- Point Addition
 - Cycle 1: $t_1 = (X_1 \cdot Z_2); t_2 = (Z_1 \cdot X_2)$
 - Cycle 1a: $M = (t_1 + t_2), Z_1 = M^2$
 - Cycle 2: $N = t_1 \cdot t_2, M = xZ_1$
 - Cycle 2a: $X_1 = M + N$

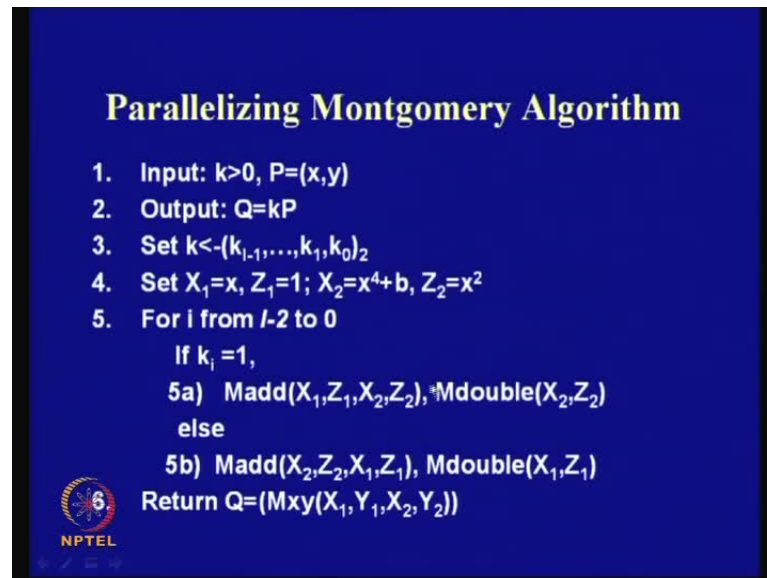


We assume that squarings and multiplications with constants can be performed without multipliers...

And you know that, from these steps, if you have got one multiplier instead of two multipliers, then, for this you need two times test; because, first you have to do this; then

you have to do this or do this, and then, you have to do this; you cannot do this two things in parallel. Similarly, here, you can do either this or you can do this; you cannot do them in parallel. So, how many clock cycles you need to do is 1, 2, 3 and 4. So, you need to do 4 clock cycles for doing this.


(Refer Slide Time: 50:46)



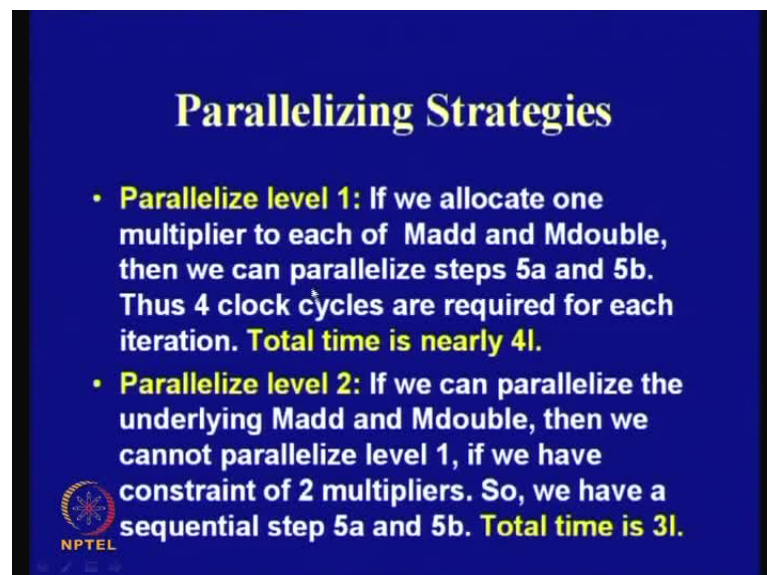
Parallelizing Montgomery Algorithm

1. Input: $k > 0, P = (x, y)$
2. Output: $Q = kP$
3. Set $k < (k_{l-1}, \dots, k_1, k_0)_2$
4. Set $X_1 = x, Z_1 = 1; X_2 = x^4 + b, Z_2 = x^2$
5. For l from $l-2$ to 0
 - If $k_l = 1$,
 - 5a) $Madd(X_1, Z_1, X_2, Z_2), *Mdouble(X_2, Z_2)$
 - else
 - 5b) $Madd(X_2, Z_2, X_1, Z_1), Mdouble(X_1, Z_1)$

Return $Q = (Mxy(X_1, Y_1, X_2, Y_2))$


 6

(Refer Slide Time: 51:10)



Parallelizing Strategies

- **Parallelize level 1:** If we allocate one multiplier to each of $Madd$ and $Mdouble$, then we can parallelize steps 5a and 5b. Thus 4 clock cycles are required for each iteration. **Total time is nearly $4l$.**
- **Parallelize level 2:** If we can parallelize the underlying $Madd$ and $Mdouble$, then we cannot parallelize level 1, if we have constraint of 2 multipliers. So, we have a sequential step 5a and 5b. **Total time is $3l$.**



And therefore, for every time step, you need to do 4 ls; therefore, if l is the number of times you are iterating this loop, you require $4l$ number of clock cycles, nearly $4l$.

(Refer Slide Time: 49:22)

Parallel Strategies for Scalar Point Multiplication

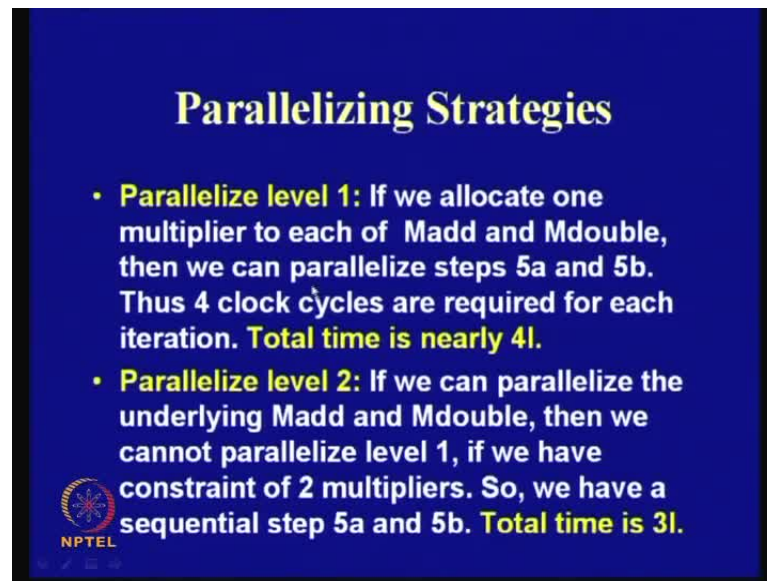
- **Point Doubling**
 - Cycle 1: $T=X_1^2, M=cZ_1^2, Z_2=T.Z_1^2$
 - Cycle 1a: $X_2=T^2+M^2$ 1 multiplier
- **Point Addition**
 - Cycle 1: $t_1=(X_1.Z_2); t_2=(Z_1.X_2)$
 - Cycle 1a: $M=(t_1+t_2), Z_1=M^2$ 2 multipliers
 - Cycle 2: $N=t_1.t_2, M=xZ_1$
 - Cycle 2a: $X_1=M+N$

NPTEL We assume that squarings and multiplications with constants can be performed without multipliers...

But suppose that, if you can, if you have got, like, if you can parallelize the underlying m add and m double, then you cannot parallelize the level 1; because you have got again a constant of two multipliers. Therefore, if you can parallelize the underlying m add and m double; that means, we are actually given two multipliers to this, right, because that is the way, how you can parallelize this. Then, you cannot do the addition and doubling simultaneously.

Because you have got a constant of two multipliers, right; because you do not have three multipliers, right. If you have three multipliers, you could have done, parallelize this as well as perform the doubling and addition in parallel, right. Now, since you have, if you have parallelized this, you have to do this and this in sequence. Which means, for this, you need two clock cycles, because, you have parallelized this and you have parallelized this. So, you basically, you can do this addition in two clock cycles, but the doubling also, we need one clock cycle.

(Refer Slide Time: 51:10)



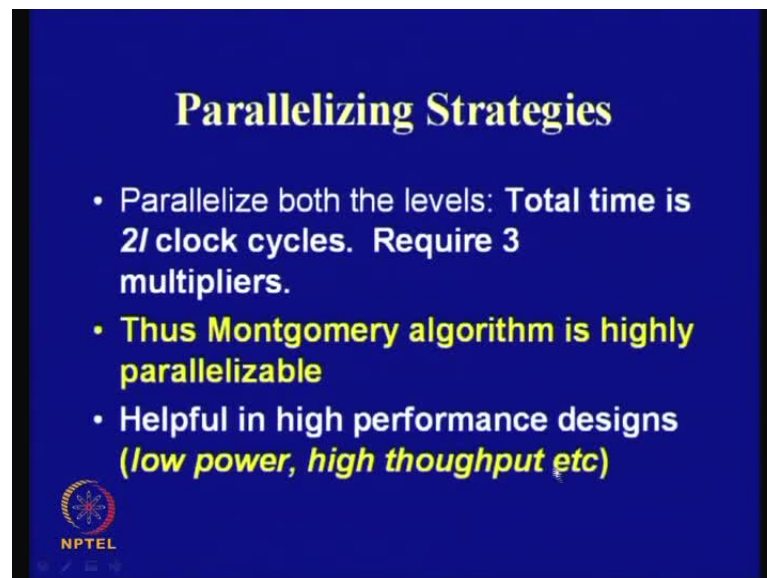
Parallelizing Strategies

- **Parallelize level 1:** If we allocate one multiplier to each of Madd and Mdouble, then we can parallelize steps 5a and 5b. Thus 4 clock cycles are required for each iteration. **Total time is nearly 4l.**
- **Parallelize level 2:** If we can parallelize the underlying Madd and Mdouble, then we cannot parallelize level 1, if we have constraint of 2 multipliers. So, we have a sequential step 5a and 5b. **Total time is 3l.**

NPTEL

And, since this doubling and addition has to be done in sequence, you need three clock cycles for doing this. And this, we will iterate, and you will require 3 l number of times for doing the entire operation.

(Refer Slide Time: 53:29)



Parallelizing Strategies

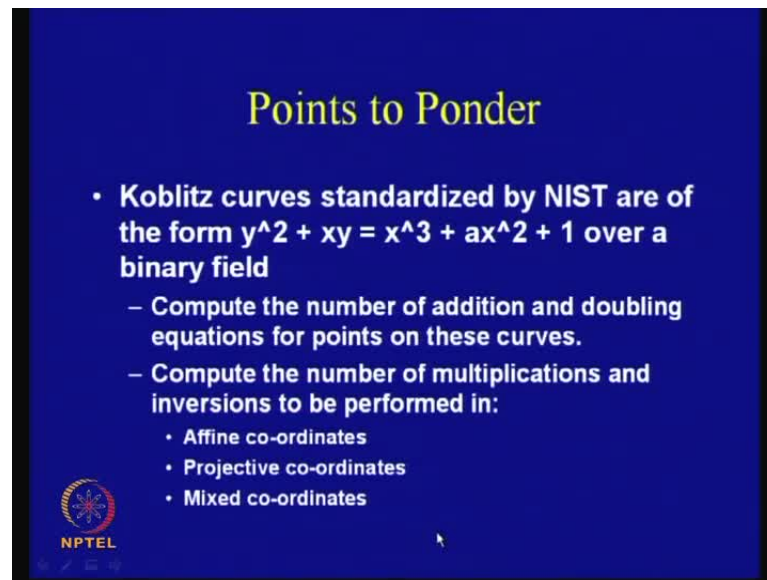
- Parallelize both the levels: Total time is 2l clock cycles. Require 3 multipliers.
- **Thus Montgomery algorithm is highly parallelizable**
- Helpful in high performance designs (*low power, high throughput etc*)

NPTEL

Suppose, you have got three multiplier, that is, you have got more resource, sorry, you can actually do this operation in lesser time. You can do that in 2 l clock cycles, sorry right; that means, you see that, Montgomery's algorithm is highly parallelizable, depending upon your constants and your requirements of power and throughput you can


actually, there is a spelling mistake here, it is throughput, therefore, you can actually make high performance designs for doing this scalar multiplications.

(Refer Slide Time: 53:58)



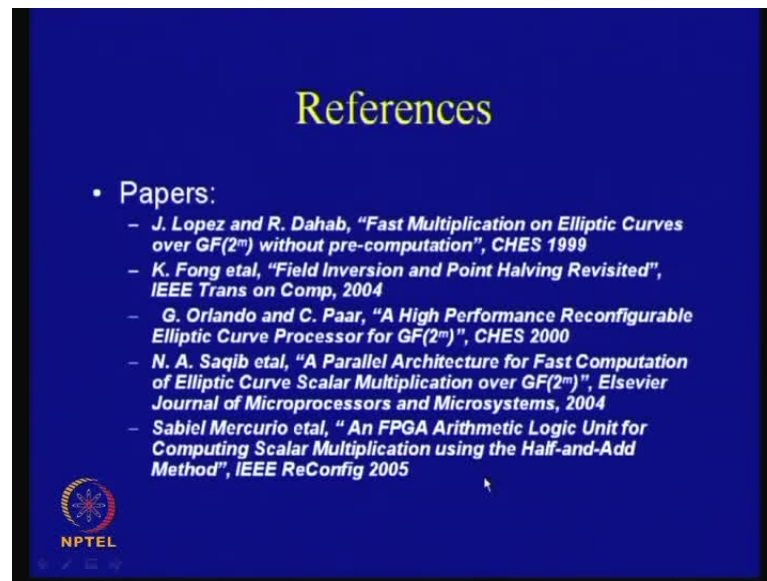
Points to Ponder

- **Koblitz curves standardized by NIST are of the form $y^2 + xy = x^3 + ax^2 + 1$ over a binary field**
 - Compute the number of addition and doubling equations for points on these curves.
 - Compute the number of multiplications and inversions to be performed in:
 - Affine co-ordinates
 - Projective co-ordinates
 - Mixed co-ordinates

 NPTEL


So, there are lot things to concentrate and think of about here. So, one of the things is, like there is something is called Koblitz curves, standardized by NIST are of the form as y squared plus xy equal to x cubed plus a x squared plus 1 over a binary field. You can take this exercise to compute the number of additions and doubling equations for points on these curves and again compute the number of multiplications and inversions to be performed in affine projective and mixed coordinates, what we have seen. So, you can rework this on specifically these kind of **cons**, where this constant a is equal to 1.

(Refer Slide Time: 54:42)



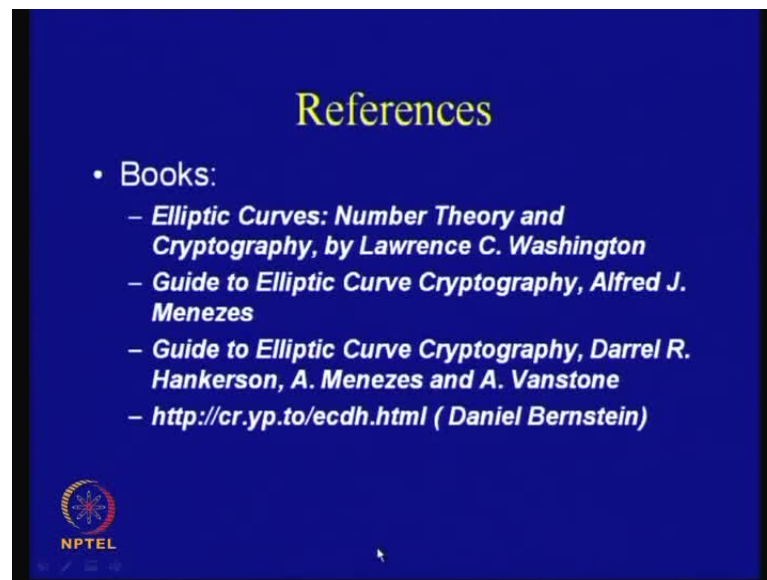
References

- Papers:
 - J. Lopez and R. Dahab, "Fast Multiplication on Elliptic Curves over $GF(2^m)$ without pre-computation", CHES 1999
 - K. Fong et al, "Field Inversion and Point Halving Revisited", IEEE Trans on Comp, 2004
 - G. Orlando and C. Paar, "A High Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$ ", CHES 2000
 - N. A. Saqib et al, "A Parallel Architecture for Fast Computation of Elliptic Curve Scalar Multiplication over $GF(2^m)$ ", Elsevier Journal of Microprocessors and Microsystems, 2004
 - Sabiel Mercurio et al, "An FPGA Arithmetic Logic Unit for Computing Scalar Multiplication using the Half-and-Add Method", IEEE ReConfig 2005


NPTEL


So, here is some of the references, some of the classic papers that I have followed are this, this Lopez and Dahab, which is published in CHES 1999 and also some of the other important references I mentioned here.

(Refer Slide Time: 54:48)



References

- Books:
 - *Elliptic Curves: Number Theory and Cryptography*, by Lawrence C. Washington
 - *Guide to Elliptic Curve Cryptography*, Alfred J. Menezes
 - *Guide to Elliptic Curve Cryptography*, Darrel R. Hankerson, A. Menezes and A. Vanstone
 - <http://cr.ypt.to/ecdh.html> (Daniel Bernstein)


NPTEL

And the books are standard this. So, next time, we will take up the topic of secret sharing schemes.