

Hardware Security
Prof. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 21
Hardware for Elliptic Curve Cryptography – III

So, welcome to this class on Hardware Security. So, today we shall be continuing with our discussions on Elliptic Curve Crypto systems and their implementations.

(Refer Slide Time: 00:27)



So, we were discussing in the last class about the Montgomery's technique of scalar multiplication. In particular today we will be discussing about how to perform a fast scalar multiplication with only one coordinate system that is only x coordinates. So, we basically do not need the y coordinates and that is a very efficient way of doing this operation.

We shall be talking about projective transformations to reduce the inversion which are very costly in terms of computations. We shall be looking into another alternative technique which is also adopted by some designers which is called as a mixed coordinate systems, where one coordinate is kept in projective coordinates whereas, the other one is kept in affine coordinates. And then finally, we shall be trying to talk about parallelization techniques. So, let us see what we can cover.

(Refer Slide Time: 01:11)

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
3. **Note that the x-coordinate of $2P$ does not depend on the y-coordinate of P**

$$x_3 = \begin{cases} \left(\frac{y_1 + y_2}{x_1 + x_2} \right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a; P \neq Q \\ x_1^2 + \frac{b}{x_1}; P = Q \end{cases}$$

$$y_3 = \begin{cases} \left(\frac{y_1 + y_2}{x_1 + x_2} \right) (x_1 + x_2) + x_1 + y_1; P \neq Q \\ x_1^2 + (x_1 + \frac{b}{x_1})x_1 + x_1; P = Q \end{cases}$$

So basically like we were discussing about this form of the curve which is defined over characteristic 2 fields where x and y are elements of $GF(2^m)$ cross $GF(2^m)$. And, we basically stopped in the last class about the by noting that the x coordinate of the doubling operation that is when we are doubling P and P that is you are computing $2P$, then the x coordinate of $2P$ depends only upon the x coordinates actually of P . So, it does not depend upon the y coordinates of P .

So, therefore, that is an important observation and we also discussed about that that we can potentially write the, or perform the computation in a manner. So, that we can only do we can also do the addition in a similar way with only the x coordinates. So, that essentially is the basic crux of Montgomery's method.

(Refer Slide Time: 01:59)

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 $P_2 = P_2 + P_1, P_1 = 2P_1$
4. Return $Q = P_1$

Invariant Property:
 $P = P_2 - P_1$

Question: How to implement the Operation efficiently?

swamyam
FREE ONLINE EDUCATION
WISDOM WAYS, LEARN WISE

And let me introduce that. So, in Montgomery's technique the objective is still to compute the scalar multiplication. So, you have got a scalar k and you would like to calculate k into P . So, k is your scalar which is say you know like the binary expansion of that is denoted by this $k_{l-1} \dots k_1 k_0$ and I want to calculate k into P . So, the technique is that we introduce 2 registers P_1 and P_2 such that P_1 is P and P_2 is $2P$ ok. So, the idea is that $P_2 - P_1$ if I calculate right $P_2 - P_1$ therefore, that is $2P - P$ ok. So, that is P ok.

So, what we try to do is we try to do this compilation in this ladder in a manner such that $2P - P$ like $P_2 - P_1$ is always equal to P . So, that remains an invariant in this computation. So, if you see like how you can do that is as follows very simple. I want to compute from again you know like assume that $l-1$ is 1. So, this is again from the MSB we are computing. So, if I compute from the MSB I am assuming $l-1$ or k_{l-1} to be 1.

So, let me try to [process/processing] start processing from k_{l-2} . So therefore, right the k_i over which I am basically processing at a current iterations like the iteration will vary from $l-2$ down to 0 . So, at some i -th stage if the k_i bit is 1 then I do an addition in the registered P_1 and I do a doubling in the register P_2 ok. Alternatively, if the bit is 0 then I do an addition in P_2 and I do a doubling in P_1 ok.

(Refer Slide Time: 04:39)

Example

Compute 7P

- $7=(111)_2$
- Initialization:
 $P_1=P; P_2=2P$
- Steps:
 - $P_1=3P, P_2=4P$
 - $P_1=7P, P_2=8P$

Compute 6P

- $7=(110)_2$
- Initialization:
 $P_1=P; P_2=2P$
- Steps:
 - $P_1=3P, P_2=4P$
 - $P_2=7P, P_1=6P$

So, if you do that then the first thing which you have observe is that $P_2 - P_1$ remains an invariant because, you see $P_2 - P_1$ is always equal to $2P_2 - P_1 - P_2$. And therefore, it does not change ok. Likewise if you calculate P if it enters this part of the loop then $P_2 - P_1$ is still equal to $P_2 + P_1 - 2P_1$ which is again equal to $P_2 - P_1$. So, it does not is still does not change ok.

So therefore, the idea is that I mean the idea is that I mean in both this computations $P_2 - P_1$ remains an invariant. And, since initialize I initialize $P_2 - P_1$ to P it remains P throughout this computation; at every stage of the iteration $P_2 - P_1$ is the base point P ok. So therefore, right what is a advantage of this? So, if I want to really understand that then we have to go across like some results that we will develop one after the other and try to see how we can do this computation ok.

The first thing is let us see the correctness before I go into that or get into the implementation aspects. So, suppose I want to calculate $7P$ again taking the same example as we were talking in the last class. So therefore, I have got 111 as my binary expansion of 7 . So, I initialize P_2 to $2P$ and P_1 to P and therefore, right if the bit is 1 I do an addition in P_1 . So, basically I add P with $2P$ and I get $3P$ and in P_2 I do a doubling.

So, P_2 becomes $4P$ ok, likewise when the bit is 1 again I mean in next iteration as well I add P I add $3P$ with $4P$. So, I get $7P$ and I double $4P$ so, I get $8P$. So, you can see that

I still get 7 P which is my objective, if the expansion was for 6 so, this will be 6 ok. So, if it was 6 then I would I I have 1 1 0. So, again I initialize P and 2 P for P 1 and P 2 respectively and since this bit is 1 I essentially do an you know like I do an addition. So, I do an addition in P 1 that is P plus 2 P is 3 P I do a doubling which is 4 P, but since is bit is 0 now I will do an addition in P 2 ok.

So therefore, I add 3 P is 4 P I get 7 P, but I double essentially P 1. So, if I double P 1 then from 3 P I get 6 P and note that P 1 is a value or the final result is stored in P 1. So, it correctly computes 6 P ok. So, you can also prove this in a very generic manner about the correctness, but intuitively we understand this algorithm works quiet ok. But, then what is a advantage if I when you are doing the computation in this manner.

(Refer Slide Time: 06:17)

The slide features a yellow background with a dark blue header and footer. The title 'Scalar Multiplication using only x-coordinates' is in a light blue font. Below the title, a red bullet point states: '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:'. The formula for x_3 is displayed in a white box with a light blue border:
$$x_3 = \frac{x_1 y_2 + x_2 y_1 + x_1 x_2^2 + x_2 x_1^2}{(x_1 + x_2)^2}$$
. Below the formula, a red-bordered box contains the hint: 'Hint: Remember that the field has a characteristic 2 and that P_1 and P_2 are points on the curve'. The footer includes the Swamyam logo and a small video feed of a man in the bottom right corner.

So, the idea is that you know like this was essentially one of the first results which actually tells you how to do elliptic curve scalar multiplication without storing any result ok. Previously like when you wanted to compute say 7 P often one technique was that you stored powers of 2 as scalars. So for example, I stored 2 P 4 P 8 P and so on and then when I am doing and computation I run time combine this results and I get return the result ok. But, in this case I can do it very efficiently just by compunations without any explicit storage and most importantly I operate in the Montgomery ladder in only 1 coordinate not in 2 coordinates ok.

So, first result that is of in significance here is this that is if P 1 is x 1 comma y 1 and P 2 is x 2 comma y 2 then the then the then the x coordinate of P 1 plus P 2 is x 3 and it can be computed by this equation. So, this essentially we can work out this result. So, this result essentially we will derive or we will essentially derive from you know like from this equation; that means, from this addition equation and of course, with some more simplifications

So, let us try to you know like do this computation. So, basically like what we essentially can do here is we can start ok.

(Refer Slide Time: 07:51)

$$\begin{aligned}
 & (x_1, y_1), (x_2, y_2) \\
 & x_3 = \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a \\
 & = \frac{y_1^2 + y_2^2 + x_1y_2 + x_2y_1 + x_2^2y_1 + x_1^2y_2 + x_1^3 + x_1^2x_2 + ax_1^2 + x_2^3 + x_2^2x_1}{(x_1 + x_2)^2} \\
 & \therefore y_1^2 + ax_1y_1 + x_1^3 + ax_1^2 + b = 0 \text{ \& } y_2^2 + ax_2y_2 + x_2^3 + ax_2^2 + b = 0. \\
 & = \frac{(x_1y_2 + x_2y_1) + (y_1^2 + x_2y_1 + x_1^3 + ax_1^2 + b) + (y_2^2 + x_1y_2 + x_2^3 + ax_2^2 + b) + (x_1^2x_2 + x_2^2x_1)}{(x_1 + x_2)^2} \\
 & = \frac{(x_1y_2 + x_2y_1) + (x_1^2x_2 + x_2^2x_1)}{(x_1 + x_2)^2}
 \end{aligned}$$

So, let us take for example, the points x 1 comma y 1 say x 1 comma y 1 and x 2 comma y 2 as the 2 points ok. So, I want to add them. So therefore, right by using the equation that we had essentially we have got x 3 equal to y 1 plus y 2 divided by x 1 plus x 2 whole square ok. I am just rewriting the equation that we had y 1 plus y 2 divided by x 1 plus x 2 plus x 1 plus x 2 plus a right.

So, now if I simplify this then in the denominator I have got x 1 plus x 2 whole square and in the numerator since, its characteristic 2 right all these elements x 1 y 1 x 2 y 2 are characteristic GF 2 power of m elements. So therefore, I will have y 1 square plus y 2 square plus this x 1 plus x 2 into y 1 plus y 2. So therefore, I can write x 1 y 1 plus x 1 y 2 plus x 2 y 1 plus x 2 y 2 plus some more terms. That is if I simplify this would be

something like $x_1^3 + x_1^2 x_2 + a x_1^2 + x_2^2 x_1 + x_2^3 + x_2^2 a$ ok. So, this is my result of the computation ok.

So, maybe we can verify this. So, note the fact that P_1 that is (x_1, y_1) and (x_2, y_2) are points on the curve ok. So, since they are point on the curve therefore, I can write that $y_1^2 + x_1 y_1 + x_1^3 + a x_1^2 + b$ is equal to 0. And, likewise $y_2^2 + x_2 y_2 + x_2^3 + a x_2^2 + b$ is equal to 0 ok.

So therefore, what we have if I write here right and arrange the terms then and keep the denominator as $(x_1 + x_2)^2$ in a numerator we will have $x_1 y_2$. So, basically keeping this terms $x_1 y_2 + x_2 y_1 + x_2 y_1$ and the remaining 2 parts are nothing, but these things. Like $y_1^2 + x_1 y_1 + x_1^3 + a x_1^2 + b + y_2^2 + x_2 y_2 + x_2^3 + a x_2^2 + b + x_1^2 + x_2^2$ ok.

So, you can note that I have just added the b here and I have also kind of adjusted the b here. So, because their characteristic 2 so, we can do that and this essentially and this part and this part essentially are both 0 ok. So therefore, right finally, I have got this result where I can write the numerator. So, the numerator is $(x_1 + x_2)^2$, but in the numerator I have got $x_1 y_2 + x_2 y_1 + x_1^2 + x_2^2$ ok. So, these is are essentially nothing, but the addition operation and is essentially that is the you know like the result that we have here and that essentially is very much this equation ok.

(Refer Slide Time: 12:29)

Scalar Multiplication using only x-coordinates

- **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:

$$x_3 = \begin{cases} x + \left(\frac{x_1}{x_1+x_2}\right)^2 + \frac{x_1}{x_1+x_2}; P_1 \neq P_2 \\ x_1^2 + \frac{b}{x_1^3}; P_1 = P_2 \end{cases}$$

Logos for Swamyam and other educational institutions are visible at the bottom of the slide.

So, now once we have this equation we can actually you know like go to the next result which essentially tells you that now with x ; remember that x comma y is suppose the point for P ok, the I mean is a point P . When you can actually like the doubling x_3 you can also write the addition x_3 only in terms of x_1 x_2 and x ok. So, you know note that x is pre determined pre is already known and therefore, the entire computation is divide of any y coordinates.

So, this result is also can also be derived quite easily and essentially we can you know like we can see this derivation also we can also do this derivation in a similar fashion ok. So, let us try to do this derivation.

(Refer Slide Time: 13:23)

$$\begin{aligned}
 & P = P_2 - P_1 \\
 & \Rightarrow (x, y) = (x_2, y_2) + (x_1, -y_1) \quad \left| \quad P_3 = P_1 + P_2 \right. \\
 & x = \frac{x_1 y_2 + x_2(x_1 + y_1) + x_1^2 + x_2^2}{(x_1 + x_2)^2} \\
 & y_3 = \frac{x_1 y_2 + x_2 y_1 + x_1^2 x_2 + x_1 x_2^2}{(x_1 + x_2)^2} \\
 & \therefore x + x_3 = \frac{x_1 x_2}{(x_1 + x_2)^2} \\
 & \therefore x_3 = x + \frac{x_1 x_2}{(x_1 + x_2)^2} = x + \frac{x_1}{x_1 + x_2} + \left(\frac{x_1}{x_1 + x_2} \right)^2
 \end{aligned}$$

So, to start with I essentially note that the point P is nothing, but P 2 minus P 1 ok. So, this point P is P 2 minus P 1 which means that x comma y is nothing, but x 2 comma y 2 plus minus P 1 ok. So, what is minus P 1? So, remember that we derive this. So, if x 1 y 1 is P 1 then minus of P 1 is nothing, but x 1 comma x 1 plus y 1. So, if this is P 1 this is your minus P 1 ok. So therefore, this is nothing, but addition with x 1 comma x 1 plus y 1 ok.

So therefore, now if I apply the previous result that we derived then I can write x as the remember it is x 1 plus x 2 whole square. But, the numerator right is x 1 y 2 plus x 2 into x 1 plus y 1 plus x 1 x 2 square plus x 1 square x 2 ok. And, also note that and we have already know that P 3 is the result of adding P 1 and P 2 that is what I am trying to derive.

So therefore, right trivially I can write that x 3 is equal to x 1 and again I can write the similar stuff for x 3 and I can write this as x 1 y 2 plus x 2 y 1 plus x 1 square x 2 plus x 1 x 2 square ok. So that means, that if I add up these 2 parts or if I add up x with x 3 then you note that all the terms gets cancelled out except x 1 x 2 divided by x 1 plus x 2 whole square ok.

So, therefore, right I can write x 3 as nothing, but x plus x 1 x 2 divided by x 1 plus x 2 whole square. So, here I mean you see that I need to do an inversion operation that is I need to calculate 1 by x 1 plus x 2, but then I need to multiply with x 2 and with x 1. So,

I need to do 2 multiplications. So, one way of kind of optimizing it further would be to write this as $x + 1$ by $x + 1$ plus $x + 2$ plus $x + 1$ divided by $x + 1$ plus $x + 2$ whole square ok. You see that if you do that, then you are doing one inversion and one multiplication with $x + 1$, but essentially it is the same computation that you are doing ok.

So therefore, this is the final form that you will take and you see again that we have completely eliminated the y coordinate and essentially are operating only on the x coordinates ok. So, with this right I mean we can now get back to the slide and we can see that essentially that is pretty much the equation which is written over here $x + 1$ by $x + 1$ plus $x + 2$ whole square plus $x + 1$ by $x + 1$ plus $x + 2$ ok. So, likewise right you can actually you know like.

(Refer Slide Time: 16:39)

Scalar Multiplication using only x-coordinates

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:

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

So, but you see that finally, when you are doing the computation you essentially have to come and final return the output point that is x_3 comma y_3 . So, you need to still compute y_3 ok. And so therefore, at end of the computation you have to essentially some way some how you know like have that information of y and using this result 3 we can do that. So, that is how it is says that if P is equal to x comma y and P_1 is x_1 comma y_1 and P_2 is equal to x_2 comma y_2 be the elliptic points ok; assume that $P_2 - P_1$ is equal to P and x is not zero ok. Then the y coordinate of P_1 can be expressed in terms of P and the x coordinates of P_1 and P_2 as follows ok.

So, you see again using the result I have obtained; so, so remember that P 1 is my final result right in the Montgomery ladder P 1 is the final result. So, therefore, the x 1 of P 1 is my important x 1 is my is the important information. What I am showing in the result is that from x 1 and from the other information like x 2 for example, you can actually pretty much compute y 1. You can you can compute y 1 and once you have x 1 and y 1 you have the coordinate you have the result basically. So, I am not going to the derivation of these results and I leave it to as an exercise to verify that this is correct ok.

(Refer Slide Time: 18:07)

Scalar Multiplication Algorithm in Affine Coordinates

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

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

- #INV: $2(i-2)+1$;
- #MULT: $2(i-2)+4$
- #ADD: $4(i-2)+6$
- #SQR: $2(i-2)+2$

So, now if I put all these things together ok; so, what you see here is that I have pretty much all the pieces now to compute the scalar multiplication algorithm in the affine coordinate system. So therefore, right this is my scalar and now what I can basically doing is I am doing is computation only using x coordinates. So, I basically initialize x 1 to remember only x 1 was supposed to initialize to P and that is equal to x. And, x 2 was supposed to initialize to 2 P and this is the x coordinate of 2 P. So therefore, I have got x square plus b by x square to initialize x 2 ok.

So, now from 1 minus 2 to 0 what are the steps that I have to do remember that if the k bit is 1, then I do an addition in P 1 and I do a doubling in P 2. So, that you can easily understand by this that is I do a computation like t equal to x 1 by x 1 plus x 2 and that is an intermediate computation that I, do stored in a temporary variable t. And then I compute x 1 which is nothing, but the x coordinate of P 1 plus P 2 ok. And therefore,

right I get x_1 plus t square plus t as my resultant x coordinate ok; x_2 stores a doubling right.

So therefore, x_2 stores a doubling of P_2 . So therefore, I can compute x_2 square plus b by x_2 square ok. Likewise if the k bit is 0 then I just do the opposite I do a doubling in x_1 . So therefore, I get x_1 equal to x_1 square plus b by x_1 square and in x_2 I calculate x plus t square plus t ok. So, note that once you have done this computation finally, you get the result which is x_1 , but somehow I need to calculate y_1 and for this I as a apply result 3. So, what I do is I calculate say x_1 plus x I calculate x_2 plus x and I denote that as r_1 and r_2 . And, then I apply result 3 to get the corresponding y coordinate and then I return x_1 comma y_1 as my resultant output.

So, note that in this computation or in this loop right I have been operating only on x coordinates, there are no y coordinates. So, basically you kind of you know like save lot of computations how to save lot of resources potentially, if you think you have a hardware design. So, let us calculate the cost of course, you can calculate the number of inversions multiplications additions and squarings which are required. Most importantly let us see about the number of inversion because, inversions as you have seen are pretty complex in in composite fields I mean in (Refer Time: 20:32) fields ok.

So therefore, right here where you see the when you are compute in number of inversion you see that there is one inversion which is require at the beginning, when you are doing in initialization. So, that is one extra inversion which you have to do. Inside the loop also you are doing 2 inversions you are always doing this inversion and you are either doing this or you are either doing this ok. So therefore, there are 2 inversions in the loop and the l minus 2 iterations in the loop.

So therefore, the number of inversions compute 2 into l minus 2 plus 1 ok. Likewise you have got multiplications additions and squarings you can essentially verify that this results are indeed correct ok. And, essentially you what you do of course, want is to reduce the number of computations. But, in a plane form a plain vanilla format this is the number of computations which you have to do ok.

(Refer Slide Time: 21:21)

Projective Co-ordinates

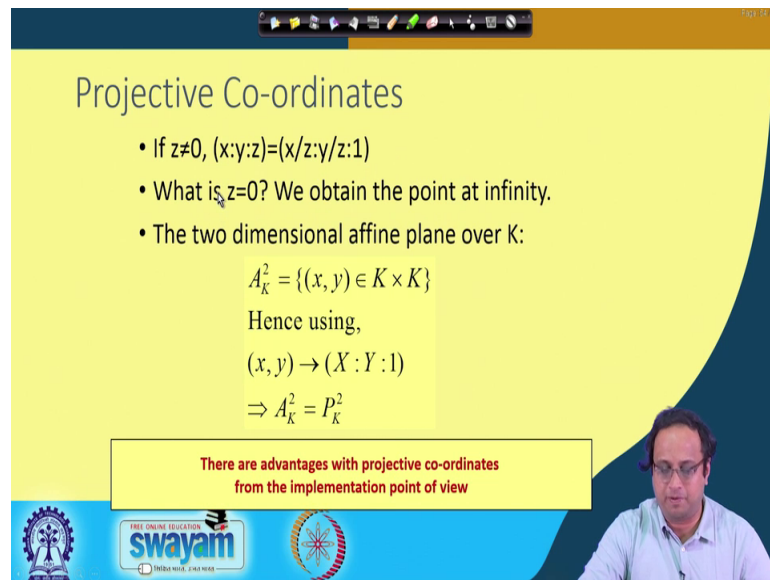
- Two-dimensional projective space P_K^2 over K is given by the **equivalence classes** of triples (x,y,z) with x,y,z in K and at least one of x, y, z nonzero.
- Two triples (x_1,y_1,z_1) and (x_2,y_2,z_2) are said to be equivalent if there exists a non-zero element λ in K , st:
 - $(x_1,y_1,z_1) = (\lambda x_2, \lambda y_2, \lambda z_2)$
 - The equivalence class depends only the ratios and hence is denoted by $(x:y:z)$

So, now there is I mean what the designers wanted to do is they wanted to reduce the number of costly inversions. Because, inversions are pretty complex and typically as a thumb rule right you will see that one inversion is equivalent to roughly 7 or at least more than 5 multiplications. So therefore, right what we would like to do is way develop techniques so, that we can optimize these computations. So, one of the techniques which is very powerful in this context is what is called as to go for a 3 coordinate system.

So, in affine coordinates what we have till now working on, we have got 2 coordinated x and y . In projective coordinates as it is called we have got 3 coordinate system so, without x y and z ok. So therefore, right what we say is that we if you are given an you know like 2 triples like $x_1 y_1 z_1$ and $x_2 y_2 z_2$, then they are said to be equivalent. If there exists a non-zero element λ which also belongs to the finite field such that x_1 comma y_1 comma z_1 is nothing, but λx_2 comma λy_2 comma λz_2 ok. So, the equivalence class is often define only by ratios like x is to y is to z ok.

So therefore, right I mean let us see about are how we can apply this technique to reduce the number of computations ok. The basic idea is that you have got a two-dimensional projective space and over K and is given by the equivalence class of triples x comma y comma z with $x y z$ in K . And, at least one of $x y z$ which is non-zero then and this is the, you know like definition that two triples are essentially equivalent. So, this is how you relate two such points ok.

(Refer Slide Time: 23:03)



Projective Co-ordinates

- If $z \neq 0$, $(x:y:z) = (x/z:y/z:1)$
- What is $z=0$? We obtain the point at infinity.
- The two dimensional affine plane over K :
 $A_K^2 = \{(x, y) \in K \times K\}$
Hence using,
 $(x, y) \rightarrow (X:Y:1)$
 $\Rightarrow A_K^2 = P_K^2$

There are advantages with projective co-ordinates from the implementation point of view

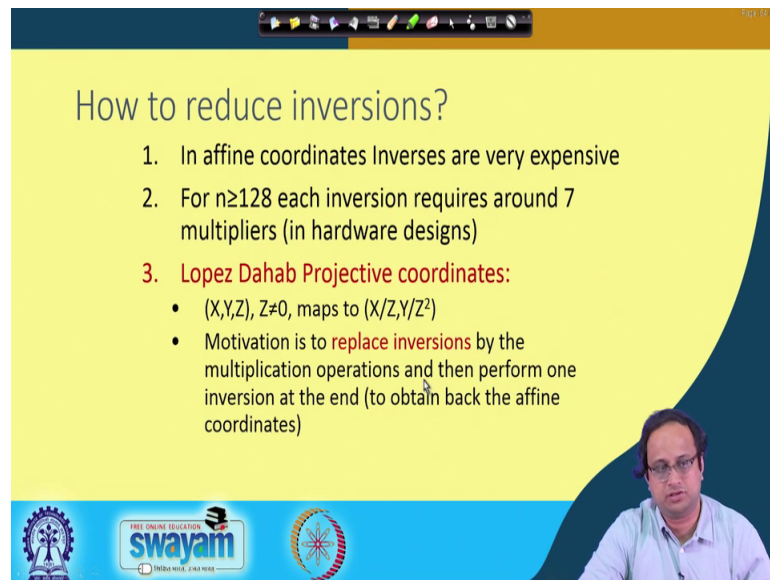
swayam

So, now I mean how do you apply them so, the that is most important. So therefore, right if so, what we do is typically what we basically take the affine coordinates like x comma y and transform that into a projective domain in the projective domain.

So, what do I do is that I basically you know like write as x by z the x coordinate and the y coordinate as y by z this is you know there are many projective coordinate systems. This is one such projective coordinate where, I right say I basically try to do is that if z is not equal to 0 then I say this ratio x is to y is to z is nothing, but x by z is to y by z is to 1 ok. If z equal to 0 right then that essentially defines the point at infinity. So, if you ever get z equal to 0 in when you are doing a projective coordinates then; that means, you have computed the point at infinity ok.

So therefore, right I mean let us try to see how we can do this computation. So, of course, like we have to you know like see the advantages and only if it is advantages because, there will be some you know like increase in computations also; when you are converting from the affine coordinates to projective coordinates what you have to see is at the increase of the cost that you are paying you whether it is giving a benefits ok. So, the idea the whole idea right whether you should go into projective coordinates will define upon your m is to y ratio; that means, the cost of the multiplier compared with your inversion circuit ok.

(Refer Slide Time: 24:33)



The slide is titled "How to reduce inversions?" and contains the following content:

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)

The slide also features a video feed of a presenter in the bottom right corner and logos for Swamyam and other institutions at the bottom.

So, so let us see how we can reduce the number of inversions any more take a more concrete case. So, in an I hope then this will be become more clear ok. So, in affine coordinates inverses are very expensive for n greater than equal to 128 each inversion requires around 7 multipliers in hardware designs ok. So, one such one popular projective coordinate system is what is called as the LD coordinate system or Lopez Dahab projective coordinates. So, what is done in this coordinates systems is that if I have got a projective coordinate like X Y and Z and Z is not equal to 0 which means is a finite point if Z equal to 0 then it is the point at infinity, then this maps to X by Z and Y by Z square ok.

So, the motivation of this is you know like of; so, now what I we what I can do is that this my relationship; that if I got a projective coordinate X Y comma Z . And, I want to transforming in to the affine coordinates then I just calculate X by Z and Y by Z square. The idea is that the motivation as I am saying is to replace inversions by multiplication operations and then perform one inversion at the end. So; that means, the entire operation will be only doing on multiplications and additions, but no inversions. But, finally, there will be 1 one inversion which is required to bring the result back from the projective coordinates to the affine coordinate systems ok.

(Refer Slide Time: 25:59)

Doubling in Projective Coordinates

• Remember:

$$x = \frac{x_1}{x_1 + x_2} + \frac{x_1}{x_1 + x_2}; P_1 \neq P_2$$

$$x_3 = \frac{x^2 + \frac{b}{x^2}}{x^2 + \frac{b}{x^2}}; P_1 = P_2$$

• In Projective Coordinates:

$P_1 = P_2, X_3 = X_1^4 + bZ_1^4$
 $Z_3 = Z_1^2 \cdot X_1^2$

$P_1 \neq P_2, Z_3 = (X_1 Z_2 + X_2 Z_1)^2$
 $X_3 = x \cdot Z_1 + (X_1 Z_2 + X_2 Z_1)(X_1 Z_1)$

Complexity Counts:

- 2 inverses
- 1 general field multiplication
- 4 additions
- 2 squarings

Optimized Complexity Counts:

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

Handwritten notes on the slide show the derivation of $x^2 + \frac{b}{x^2}$ in projective coordinates as $\frac{X^2 + bZ^2}{Z^2}$ and its conversion to $X_3 = X_1^4 + bZ_1^4$ and $Z_3 = Z_1^2 X_1^2$ for the case $P_1 = P_2$.

So, let us you look at the equations to understand this. For example, this was my equation which we essentially derived for doing the doubling in the projective coordinates ok. So, if you remember what we so, let us take then easy example when we are considering the doubling for example, ok. And, the similar thing can be done for addition as well which I can which I leave for you to as an exercise.

So, what so, let us see this equation x^3 equal to x^1 square plus b by x^1 square. You can easily see that if I want to do this computation I need to do one inversion operation because, I have to do 1 by x^1 I have to calculate this 1 by x^1 . So now, let us see how we can transform this into the projective coordinate system. So, the idea is that what we do is we have to do this computation. So, the computation that is require to be performed is x^1 square plus b by x^1 square ok.

So, now my projective my computation that I said using the Lopez Dahab right is X by Z Y by Z square. This is my you know like if I have got a point say x y z , then and if I want to write that in the affine coordinates then my representation is X by Z and Y by Z square. So therefore, right if I just take this and I plug in to this equation then what do I have, I have got X by Z whole square. So, I am just writing from this X by Z whole square plus b divided by X by Z whole square right. So, that implies that this is nothing, but Z square X square in the denominator. And, the numerator is nothing, but X to the

power of 4 plus bZ to the power of 4 right. So, this is your corresponding result in the you know like in the affine coordinates.

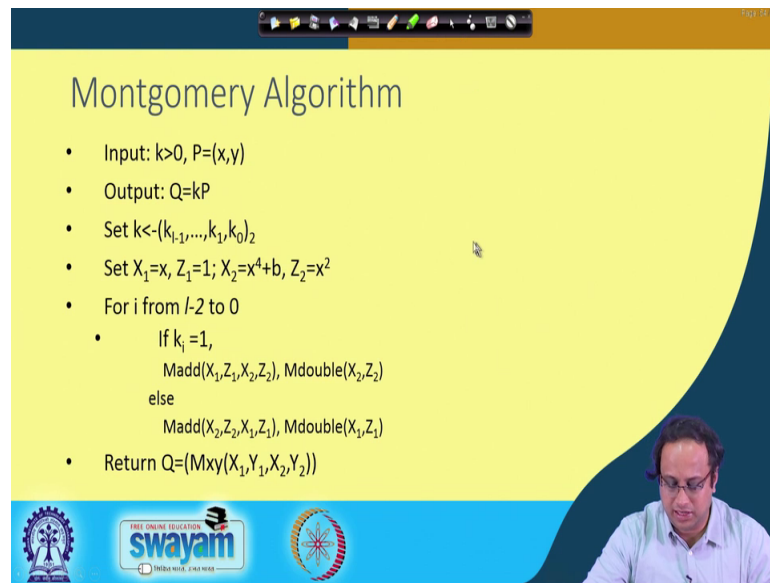
So therefore, right what we will do here is we will actually calculate; so, so now, imagine that the computation is done in the projective domain ok. So, in the projective domain you are so, in so, in a basically; so, if you compare it with the affine domain what you are doing is, in the affine domain you have got x_1 comma y_1 . So, actually you have got x_1 in this case which is required. You are basically calculating x_3 from there right; this is the affine domain. But, in the projective domain now you have got X_1 comma Y_1 comma Z_1 from there you are calculating X_3 and Z_3 . Actually you know like as I said that we do not need Y_1 , because of the because of the advantages of Montgomery ladder. We are basically operating on $X_1 Z_1$ and calculating X_3 and Z_3 ok.

So therefore, now if you compare this consider this as your X_3 . So, then that implies because of the you know like if you see the Lopez Dahab representation then this stands for X_3 and this stands on the denominator stands for Z_3 . So that means, now I can process the numerator and the denominator separately ok. So, this is essentially nothing, but my X_3 equation and this is my Z_3 question ok. So, likewise right in even if you see the addition operation when you are doing addition, you also see that you are you can you can process the X_3 and the Z_3 separately ok. So, there are no inversions which are required. So of course, if you compare you see that there are 0 inverses which were required ok.

So, if you if you see that this is you are cost for doing the doubling and the addition in the normal affine coordinates. In the projective coordinates you see that there is an increase in the number of multiplication. For example, you are going from 1 to 4 ok, but what you are saving is this costly inverse. So therefore, the advantage of going for affine to projective coordinates in only come if your inverse circuit is complex ok.

So therefore, for smaller fields this may not be so advantageous, but for normal fields or the fields of you know usual dimensions in elliptic curves you have pretty large or reasonably large you will find that this should be advantages technique. So therefore, right what we will see in our future discussions is that whenever we are trying to do and hardware develop an hardware design usually we use the projective coordinate system because, you want to reduce the number of inversions.

(Refer Slide Time: 30:25)



Montgomery Algorithm

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

swayam
INDIA WISE

So finally, right I mean if you when you when you get the result. So now, you can essentially do the Montgomery ladder in entirely in the projective coordinates. So, essentially I have done exactly nothing, but I just written X_1 equal to x . So, you see I am just processing only on X and Z 's because I do not need the Y coordinates. Again if it is just you know like you have return the X_2 to stored the doubling operation and that is exactly the equations that we derived in the last slide. And, then in the iteration or in the loop we basically do an addition in $X_1 Y_1 X_2 Z_2$ and do the doubling.

So, these are all projective coordinate additions and doublings and likewise you are doing addition and doubling in X to Z_2 and in $X_1 Z_1$ or doubling only in $X_1 Z_1$ if the key bit is 0 ok. So finally, right but finally, right when you get the result you have got $X_1 Y_2$ and $X_2 Y_2$, but you need to again bring back that to the affine coordinate system ok.

(Refer Slide Time: 31:17)

Mxy: Projective to Affine

$$x_3 = X_1 / Z_1$$
$$y_3 = (x + X_1^2 / Z_1) / ((X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2))(xZ_1Z_2)^{-1} + y$$

Requires 10 multiplications and one inverse operation

swamyam
FREE ONLINE EDUCATION
IIT BOMBAY

And, there you essentially have to do one more inversion because here you see that I essentially need to bring the result back to the you know like from the projective coordinate system to the affine coordinate system. So, I need to do this X_1 by Z_1 and that is easy to do, but in the y_3 right do few more computations ok. So, the y_3 is again you know like a manipulation that you have to do depending upon the equation that I show to you in result 3, which essentially will bring you back the result ok. And finally, right you will that this entire operation will require 10 multiplications and one inversion ok.

So, now, you know like I I basically you know like leave it to you as an exercise to think that why you need one inverse ok, because you see that it looks like you have to do a 1 by Z_1 here and a different inversion here ok. So, I just leave it to you as an exercise to think about how or why you basically need one inversion to do this operation ok.

(Refer Slide Time: 32:17)

<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

Log 32:17

swamyam

So therefore, right let us try to finally, make a comparison of the affine coordinates with the projective coordinates just to see the benefits ok. So, you see that in the affine coordinates these are the rough nomenclature of the you know as I said that you have to do 2 into $\lceil \log k \rceil$, $2 \log k$ is typically denoting the dimension of the bits ok. So, we derive that it was $2 \log k - 2 + 1$ or $2 \log k - 1$ and that is basically is written here as $2 \log k + 1$, kind of denoting the number of inversion which you have to do. Interesting in the projective coordinates you have to do only one inversion and that is the benefit, but at least you can also understand that there is the cost. The cost is in terms of multiplications, the multiplications as increased you know like.

So therefore, right the question is that whether the final decision whether you will go into the projective coordinates will depend upon your, I is to M ratio that is the inversion is to the multiplication ratio. Typically right if it is more than 5 like, if it is an around 7 then we go into the projective coordinate systems ok.

(Refer Slide Time: 33:17)

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.

$$U = Z_2^2 Y_1 + Y_2, S = Z_2 X_1 + X_2, T = Z_2 S, Z_3 = T^2,$$
$$V = Z_3 X_1, X_3 = U^2 + T(U + S^2 + Ta),$$
$$Y_3 = (V + X_3)(TU + Z_3) + Z_3^2 C$$

**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 %...**

swamyam
FREE ONLINE EDUCATION
TRIED AND TRUE

So finally, there is you know like there is a technique also which is called as a mix coordinate systems. In mix coordinate systems what you do is you basically keep 1 point in the affine coordinates whereas, the other point is which is suppose you have got 2 points P_1 which is X_1 comma by Z_1 Y_1 by Z_1 squared. And, P_2 is X_2 by Z_2 and Y_2 by Z_2 square be two points on the curve ok, if Z_1 is 1, if Z_1 is 1 implies the this is this point is an affine coordinate point. So, if P_1 is affine and P_2 is projective then also you can calculate P_1 plus P_2 ok. And, here are some equations to do that and I am not getting into the derivation of these equations. But, what I would like to mention here is that there are some papers or some results which shows the number of multiplications are further reduced ok.

Most importantly squaring is increased a bit, but they are cheap in $GF(2^n)$ as we have seen. So, so, roughly that the result; so, that if a is not equal to 0 then there is an improvement of 10 percent and if a is equal to 0 then there is an improvement of 12 percent ok. So therefore, it makes also sense to keep one point in affine coordinate and the other point in projective coordinates and to do the computations ok.

So, let me stop here and we shall continue with some more discussions about how we can parallelize it and eventually go in to the hardware design of elliptic curves in the next class.

So, thank you for your attention.