Elliptic curve scalar multiplication combining Yao's algorithm and the double base number system

Nicolas Méloni and M. Anwar Hasan

Department of Electrical and Computer Engineering
University of Waterloo

May 12th, 2009

Outline

Elliptic curve scalar multiplication

Elliptic curve over a finite fields: $\mathbb{K}$ a characteristic p field (with $p > 3$), $E(\mathbb{K})$ the set of points $(x, y) \in \mathbb{K}^2$ satisfying $y^2 = x^3 + ax + b$, with $a, b \in \mathbb{K}$.

Point scalar multiplication: P a point on E and $k > 0$, compute $[k]P = P + \cdots + P$ (k times).

How to perform this operation as fast as possible?

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P \rightarrow 3P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \to 2P \to 3P \to 6P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P \rightarrow 3P \rightarrow 6P \rightarrow 7P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P \rightarrow 3P \rightarrow 6P \rightarrow 7P \rightarrow 14P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P \rightarrow 3P \rightarrow 6P \rightarrow 7P \rightarrow 14P \rightarrow 28P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P \rightarrow 3P \rightarrow 6P \rightarrow 7P \rightarrow 14P \rightarrow 28P \rightarrow 29P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P \rightarrow 3P \rightarrow 6P \rightarrow 7P \rightarrow 14P \rightarrow 28P \rightarrow 29P$

Non-Adjacent Form (NAF)

$k = \sum_{i=0}^{n-1} k_i 2^i, k_i \in \{-1, 0, 1\}$, n doublings and $n/3$ additions (on average)
$NAF(29) = 2^5 - 2^2 + 2^0 = 100\bar{1}01_2$
$P \rightarrow 2P \rightarrow 4P \rightarrow 8P \rightarrow 7P \rightarrow 14P \rightarrow 28P \rightarrow 29P$

Scalar multiplication algorithms

Double-and-Add

$k = \sum_{i=0}^{n-1} k_i 2^i$, $n-1$ doublings and $n/2$ additions (on average)
$k = 29 = 2^4 + 2^3 + 2^2 + 2^0 = 11101_2$
$P \rightarrow 2P \rightarrow 3P \rightarrow 6P \rightarrow 7P \rightarrow 14P \rightarrow 28P \rightarrow 29P$

Non-Adjacent Form (NAF)

$k = \sum_{i=0}^{n-1} k_i 2^i, k_i \in \{-1, 0, 1\}$, n doublings and $n/3$ additions (on average)
$NAF(29) = 2^5 - 2^2 + 2^0 = 100\bar{1}01_2$
$P \rightarrow 2P \rightarrow 4P \rightarrow 8P \rightarrow 7P \rightarrow 14P \rightarrow 28P \rightarrow 29P$

wNAF

$k = \sum_{i=0}^{n-1} k_i 2^i, |k_i| < 2^{w-1}$ n doublings , $n/(w+1)$ additions (on average) and precomputations
$3NAF(29) = 2^5 - 3 \times 2^0 = 10000\bar{3}_2$
$P \rightarrow 2P \rightarrow 4P \rightarrow 8P \rightarrow 16P \rightarrow 32P \rightarrow 29P$

Yao's algorithm

Let $k = k_{n-1}2^{n-1} + \cdots + k_1 2 + k_0$ with $k_i \in \{0, 1, 3, \ldots, 2^w - 1\}$

- Compute $2^i P \; \forall i \leq n-1$
- $d(1)P, \ldots, d(2^w - 1)P$, where $d(j)$ is the sum of the 2^i such that $k_i = j$
- kP is obtained as $d(1)P + 3d(3)P + \cdots + (2^w - 1)d(2^w - 1)P$

It is equivalent to rewrite k as:

$$k = 1 \times \underbrace{\sum_{k_i=1}}_{d(1)} 2^i + 3 \times \underbrace{\sum_{k_i=3}}_{d(3)} 2^i + \cdots + (2^w - 1) \times \underbrace{\sum_{k_i=2^w-1}}_{d(2^w-1)} 2^i$$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute
- $d(1)P =$
- $d(3)P =$
- $d(5)P =$
- $d(7)P =$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute P
- $d(1)P =$
- $d(3)P =$
- $d(5)P =$
- $d(7)P = P$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3 P$
- $d(1)P =$
- $d(3)P =$
- $d(5)P = 2^3 P$
- $d(7)P = P$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3 P \ldots 2^8 P$
- $d(1)P =$
- $d(3)P = 2^8 P$
- $d(5)P = 2^3 P$
- $d(7)P = P$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3 P \ldots 2^8 P \ldots 2^{11} P$
- $d(1)P = 2^{11}P$
- $d(3)P = 2^8 P$
- $d(5)P = 2^3 P$
- $d(7)P = P$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \dots 2^3 P \dots 2^8 P \dots 2^{11} P \dots 2^{14} P$
- $d(1)P = 2^{11} P$
- $d(3)P = 2^8 P + 2^{14} P$
- $d(5)P = 2^3 P$
- $d(7)P = P$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3 P \ldots 2^8 P \ldots 2^{11} P \ldots 2^{14} P \ldots 2^{18} P$
- $d(1)P = 2^{11}P + 2^{18}P$
- $d(3)P = 2^8 P + 2^{14}P$
- $d(5)P = 2^3 P$
- $d(7)P = P$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3 P \ldots 2^8 P \ldots 2^{11} P \ldots 2^{14} P \ldots 2^{18} P$
- $d(1)P = 2^{11}P + 2^{18}P$
- $d(3)P = 2^8 P + 2^{14} P$
- $d(5)P = 2^3 P$
- $d(7)P = P$
- $kP = 7d(7)P + 5d(5)P + 3d(3)P + d(1)P$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times 2^{18} + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \dots 2^3 P \dots 2^8 P \dots 2^{11} P \dots 2^{14} P \dots 2^{18} P$
- $d(1)P = 2^{11}P + 2^{18}P$
- $d(3)P = 2^8 P + 2^{14} P$
- $d(5)P = 2^3 P$
- $d(7)P = P$
- $kP = 7d(7)P + 5d(5)P + 3d(3)P + d(1)P$

Same number of operations as the previous methods.

Double-base number system

Definition

$k \geq 0,\ k = \sum_{i=1}^{n} 2^{b_i} 3^{t_i}$

Properties

- Such a representation always exists
- It is highly redundant: 127 has 783 representations!
- Some of them are very sparse (canonical representation)

Example

- $127 = 2^2 3^3 + 2^1 3^2 + 2^0 3^0 = 108 + 18 + 1$
- 431 is the smallest integer requiring four summands
- 18431, 3448733 and 1441896119 are the smallest integers requiring, respectively, five, six and seven summands

Considered as not really efficient for scalar multiplication purposes.

Double-base chains

Definition

Given $k > 0$, a sequence $(C_i)_i > 0$ of positive integers satisfying:
$C_1 = 1,\ C_{i+1} = 2^{b_i} 3^{t_i} C_i + d_i$, with $d_i \in \{-1, 1\}$
for some $b_i, t_i \geq 0$ and such that $C_n = k$ for some n is called a double-base chain computing k.

Example

- $k = 1717 = 2^6 3^3 + 2^2 3 + 1$
- $kP = 2^2 3(2^4 3^2 P + P) + P$
- $2^4 3^2 P \rightarrow 2^4 3^2 P + P \rightarrow 2^6 3^3 P + 2^2 3P \rightarrow 2^6 3^3 P + 2^2 3P + P$
- 6 doublings and 3 triplings

Yao's algorithm adapted to double-base number system

$k = 2^{b_n}3^{t_n} + \cdots + 2^{b_1}3^{t_1}$

- Compute $2^i P \; \forall i \leq b_{max} = \max_i(b_i)$
- For all $j \leq t_{max}$, compute $d(0)P, d(1)P, \ldots, d(t_{max})P$, where $d(j)$ is the sum of the 2^i such that $t_i = j$
- kP is obtained as: $d(0)P + 3d(1)P + 3^2 d(2)P + \cdots + 3^{t_{max}} d(t_{max})P$

It is equivalent to rewrite k as:

$$k = \underbrace{\sum_{t_i=0} 2^i}_{d(0)} + 3 \times \underbrace{\sum_{t_i=1} 2^i}_{d(1)} + 3^2 \times \underbrace{\sum_{t_i=2} 2^i}_{d(2)} + \cdots + 3^{t_{max}} \times \underbrace{\sum_{t_i=t_{max}} 2^i}_{d(t_{max})}$$

Yao's algorithm adapted to double-base number system

Let $k = 314159 = 2^{10}3^5 + 2^8 3^5 + 2^{10}3^1 + 2^2 3^2 + 3^2 + 2^1 3^0$
$\mathtt{max}(a_i) = 10$ and $\mathtt{max}(b_i) = 5$:

- Compute
- $d(0)P =$
- $d(1)P =$
- $d(2)P =$
- $d(5) =$

Yao's algorithm adapted to double-base number system

Let $k = 314159 = 2^{10}3^5 + 2^8 3^5 + 2^{10}3^1 + 2^2 3^2 + 3^2 + 2^1 3^0$
$\mathtt{max}(a_i) = 10$ and $\mathtt{max}(b_i) = 5$:

- Compute P
- $d(0)P =$
- $d(1)P =$
- $d(2)P = P$
- $d(5) =$

Yao's algorithm adapted to double-base number system

Let $k = 314159 = 2^{10}3^5 + 2^8 3^5 + 2^{10}3^1 + 2^2 3^2 + 3^2 + 2^1 3^0$
$\mathtt{max}(a_i) = 10$ and $\mathtt{max}(b_i) = 5$:

- Compute $P \ldots 2P$
- $d(0)P = 2P$
- $d(1)P =$
- $d(2)P = P$
- $d(5) =$

Yao's algorithm adapted to double-base number system

Let $k = 314159 = 2^{10}3^5 + 2^8 3^5 + 2^{10}3^1 + 2^2 3^2 + 3^2 + 2^1 3^0$
$\mathtt{max}(a_i) = 10$ and $\mathtt{max}(b_i) = 5$:

- Compute $P \ldots 2P \ldots 2^2 P$
- $d(0)P = 2P$
- $d(1)P =$
- $d(2)P = P + 2^2 P$
- $d(5) =$

Yao's algorithm adapted to double-base number system

Let $k = 314159 = 2^{10}3^5 + 2^8 3^5 + 2^{10}3^1 + 2^2 3^2 + 3^2 + 2^1 3^0$
$\mathtt{max}(a_i) = 10$ and $\mathtt{max}(b_i) = 5$:

- Compute $P \dots 2P \dots 2^2 P \dots 2^8 P$
- $d(0)P = 2P$
- $d(1)P =$
- $d(2)P = P + 2^2 P$
- $d(5) = 2^8 P$

Yao's algorithm adapted to double-base number system

Let $k = 314159 = 2^{10}3^5 + 2^8 3^5 + 2^{10}3^1 + 2^2 3^2 + 3^2 + 2^1 3^0$
$\max(a_i) = 10$ and $\max(b_i) = 5$:

- Compute $P \ldots 2P \ldots 2^2 P \ldots 2^8 P \ldots 2^{10} P$
- $d(0)P = 2P$
- $d(1)P = 2^{10} P$
- $d(2)P = P + 2^2 P$
- $d(5) = 2^8 P + 2^{10} P$

Yao's algorithm adapted to double-base number system

Let $k = 314159 = 2^{10}3^5 + 2^8 3^5 + 2^{10}3^1 + 2^2 3^2 + 3^2 + 2^1 3^0$
$\mathtt{max}(a_i) = 10$ and $\mathtt{max}(b_i) = 5$:

- Compute $P \dots 2P \dots 2^2 P \dots 2^8 P \dots 2^{10} P$
- $d(0)P = 2P$
- $d(1)P = 2^{10} P$
- $d(2)P = P + 2^2 P$
- $d(5) = 2^8 P + 2^{10} P$
- $kP = 3^5 d(5)P + 3^2 d(2)P + 3d(1)P + d(0)P$
- $= 3(3(3^3 d(5]P + d(2)P) + d(1)P) + d(0)P$

Yao's algorithm adapted to double-base number system

Remarks

- We choose some bound over b_{max} and t_{max} so that $2^{b_{max}}3^{t_{max}} \sim k$
- Less restrictive than double-base chain approach

Comparison with double-base chains

- Same number of doublings and triplings (because of the bounds)
- Lower number of additions

Caching strategies

A point P is represented as $(X : Y : Z)$.

After an addition

A point addition involving P requires the computation of Z^2 and Z^3, one usually caches those data to decrease the cost a new addition involving P (from 11M+5S to 10M+4S).

After an doubling

Doubling P requires the computation of Z^2, One can caches those data to decrease the cost a new addition involving P (from 11M+5S to 11M+4S).

The second case never happens *chain based* algorithms, a point is never reused after being duplicated.

Caching strategies

- **addition after doubling (dADD)**: addition of a point that has already been doubled before
- **double addition after doubling (2dADD)**: addition of two points that have already been doubled before
- **addition after doubling + readdition (dreADD)**: addition of a point that has already been doubled before to a point that has been added before
- **addition after doubling + mixed addition dmADD**: addition of a point that has already been doubled before to a point in affine coordinate (i.e. $Z = 1$)

Caching strategies

Curve shape	ADD	dADD	2dADD	dreADD	dmADD
3DIK	11M+6S	11M+6S	11M+6S	10M+6	7M+4S
Edwards	10M+1S	10M+1S	10M+1S	10M+1S	9M+1S
ExtJQuartic	7M+4S	7M+3S	7M+2S	7M+2S	6M+2S
Hessian	6M+6S	6M+6S	6M+6S	6M+6M	5M+6S
InvEdwards	9M+1S	9M+1S	9M+1S	9M+1S	8M+1S
JacIntersect	11M+1S	11M+1S	11M+1S	11M+1S	10M+1S
Jacobian	11M+5S	11M+4S	11M+3S	10M+3S	7M+3S
Jacobian-3	11M+5S	11M+4S	11M+3S	10M+3S	7M+3S

Table: New elliptic curve operations cost

Performing tests

Methodology

For 160-bit scalars and all values of b_{max} and t_{max} such that $2^{b_{max}}3^{t_{max}}$ is 160-bit integer.

For each curve and each set of parameters, we have:

- generated 1000 pseudo random integers in $\{0, \ldots, 2^{160} - 1\}$,
- converted each integer into DBNS using the corresponding parameters,
- counted all the operations involved in the point scalar multiplication process.

Performing tests

Curve shape	DBL	TPL	ADD	reADD	dADD	2dADD
3DIK	43.50	73.43	1.20	0.64	16.10	3.49
Edwards	139.12	12.84	1.68	1.55	18.48	0.97
ExtJQuartic	139.12	12.84	1.68	1.55	18.48	0.97
Hessian	112.22	29.73	1.26	1.07	17.40	1.63
InvEdwards	139.12	12.84	1.68	1.55	18.48	0.97
JacIntersect	142.19	10.94	2.40	1.64	17.71	0.81
Jacobian	130.10	18.71	1.43	1.09	18.36	1.11
Jacobian-3	130.10	18.71	1.43	1.09	18.36	1.11

2reADD	dreADD	mADD	dmADD	mreADD
0.01	0.29	0.66	0.45	0.01
0	0.01	1.59	0.22	0.01
0	0.01	1.59	0.22	0.01
0.01	0.17	1.07	0.28	0.03
0	0.01	1.59	0.22	0.01
0	0.13	2.22	0.29	0.03
0	0.14	1.31	0.25	0.03
0	0.14	1.31	0.25	0.03

Table: Detailed operation count for the Yao-DBNS scalar multiplication using 160-bit scalar

Curve shape	Method	b_{max}	t_{max}	# group operations
3DIK	DB chain[1]	80	51	1502.4
	Yao-DBNS	44	74	1477.3
Edwards	DB chain	156	3	1322.9
	Yao-DBNS	140	13	1283.3
ExtJQuartic	DB chain	156	3	1311.0
	(2,3,5)NAF[2]	131	12	1226.0
	Yao-DBNS	140	13	1210.9
InvEdwards	DB chain	156	3	1290.3
	(2,3,5)NAF	142	9	1273.8
	Yao-DBNS	140	13	1258.6
Jacobian-3	DB chain	100	38	1504.3
	(2,3,5)NAF	131	12	1426.8
	Yao-DBNS	131	19	1475.3

Table: Optimal parameters and operation count for 160-bit scalars

[1] D. J. Bernstein and P. Birkner and T. Lange and C. Peters, *Optimizing Double-Base Elliptic-Curve Single-Scalar Multiplication* , 2007

[2] P. Longa and C. Gebotys, *Setting Speed Records with the (Fractional) Multibase Non-Adjacent Form Method for Efficient Elliptic Curve Scalar Multiplication*, 2009

Conclusions

- Yao-DBNS algorithm is less restrictive than double-base chains and faster
- Works with any other double base system
- What about multi-base number systems ...
- ... and multi-scalar multiplication ?

Conclusions

- Yao-DBNS algorithm is less restrictive than double-base chains and faster
- Works with any other double base system
- What about multi-base number systems ...
- ... and multi-scalar multiplication ?

Thanks

nmeloni@vlsi.uwaterloo.ca
http://www.vlsi.uwaterloo.ca/~nmeloni/