Elliptic Curve Point Multiplication Combining Yao's Algorithm and Double Bases

Nicolas Méloni and M. Anwar Hasan

Department of Electrical and Computer Engineering
University of Waterloo

September 8th, 2009

Outline

How to perform $[k]P = P + P + \cdots + P$?

Double-and-add

- $k = \sum_{i=0}^{n-1} k_i 2^i$
- $k = 267 = 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1 \rightarrow 267P = 2(2(2^2(2^5P + P) + P) + P) + P$
- 1 doubling per bit + 1 addition per non-zero bit

Non Adjacent Form (NAF)

- $k_i \in \{-1, 0, 1\}$
- $k = 31 = 11111_2 = 10000\bar{1}_{NAF}$
- at most n doublings and $\frac{n}{3}$ additions (on average)

How to perform $[k]P = P + P + \cdots + P$?

Brauer's Algorithm (wNAF)

- $|k_i| < 2^{w-1}$
- $k = 267 = 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1$
- 2-NAF: $1\ 0\ 0\ 0\ 1\ 0\ \overline{1}\ 0\ \overline{1}$
- 3-NAF: $1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 3$
- 4-NAF: $1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ \overline{5}$
- 5-NAF: $1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 11$

Two steps

- precompute $3P, \ldots, (2^w - 1)P$
- perform the horner scheme

Yao's algorithm

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

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 $2^i P \ \forall i \leq n - 1$

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 $k_i = j$

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 $d(1)P + 3d(3)P + \cdots + (2^w - 1)d(2^w - 1)P$

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:

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

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^{8}) + 5 \times 2^3 + 7 \times 2^0$

- Compute
- $d(1)P \quad = 000\,0000\,0000\,0000\,0000$
- $d(3)P \quad = 000\,0000\,0000\,0000\,0000$
- $d(5)P \quad = 000\,0000\,0000\,0000\,0000$
- $d(7)P \quad = 000\,0000\,0000\,0000\,0000$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute P
- $d(1)P = 000\,0000\,0000\,0000\,0000$
- $d(3)P = 000\,0000\,0000\,0000\,0000$
- $d(5)P = 000\,0000\,0000\,0000\,0000$
- $d(7)P = 000\,0000\,0000\,0000\,0001$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^{8}) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \dots 2^3 P$
- $d(1)P \quad = 000\,0000\,0000\,0000\,0000$
- $d(3)P \quad = 000\,0000\,0000\,0000\,0000$
- $d(5)P \quad = 000\,0000\,0000\,0000\,1000$
- $d(7)P \quad = 000\,0000\,0000\,0000\,0001$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.

$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^{8}) + 5 \times 2^{3} + 7 \times 2^{0}$

- Compute $P \ldots 2^3 P \ldots 2^8 P$
- $d(1)P \quad = 000\,0000\,0000\,0000\,0000$
- $d(3)P \quad = 000\,0000\,0001\,0000\,0000$
- $d(5)P \quad = 000\,0000\,0000\,0000\,1000$
- $d(7)P \quad = 000\,0000\,0000\,0000\,0001$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3 P \ldots 2^8 P \ldots 2^{11} P$
- $d(1)P = 000\,0000\,1000\,0000\,0000$
- $d(3)P = 000\,0000\,0001\,0000\,0000$
- $d(5)P = 000\,0000\,0000\,0000\,1000$
- $d(7)P = 000\,0000\,0000\,0000\,0001$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^{8}) + 5 \times 2^{3} + 7 \times 2^{0}$

- Compute $P \dots 2^3 P \dots 2^8 P \dots 2^{11} P \dots 2^{14} P$
- $d(1)P \quad = 000\,0000\,1000\,0000\,0000$
- $d(3)P \quad = 000\,0100\,0001\,0000\,0000$
- $d(5)P \quad = 000\,0000\,0000\,0000\,1000$
- $d(7)P \quad = 000\,0000\,0000\,0000\,0001$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3 P \ldots 2^8 P \ldots 2^{11} P \ldots 2^{14} P \ldots 2^{18} P$
- $d(1)P \quad = 100\,0000\,1000\,0000\,0000$
- $d(3)P \quad = 000\,0100\,0001\,0000\,0000$
- $d(5)P \quad = 000\,0000\,0000\,0000\,1000$
- $d(7)P \quad = 000\,0000\,0000\,0000\,0001$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3P \ldots 2^8P \ldots 2^{11}P \ldots 2^{14}P \ldots 2^{18}P$
- $d(1)P \quad = 100\,0000\,1000\,0000\,0000$
- $3 \times d(3)P \quad = 000\,0100\,0001\,0000\,0000$
- $5 \times d(5)P \quad = 000\,0000\,0000\,0000\,1000$
- $7 \times d(7)P \quad = 000\,0000\,0000\,0000\,0001$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^{8}) + 5 \times 2^{3} + 7 \times 2^{0}$

- Compute $P \ldots 2^3 P \ldots 2^8 P \ldots 2^{11} P \ldots 2^{14} P \ldots 2^{18} P$
- $d(1)P = 100\,0000\,1000\,0000\,0000$
- $3 \times d(3)P = 000\,0300\,0003\,0000\,0000$
- $5 \times d(5)P = 000\,0000\,0000\,0000\,5000$
- $7 \times d(7)P = 000\,0000\,0000\,0000\,0007$

Yao's algorithm

Example

Let $k = 314159 = 100\,0300\,1003\,0000\,5007$, $n = 19$ and $2^w - 1 = 7$.
$k = 1 \times (2^{18} + 2^{11}) + 3 \times (2^{14} + 2^8) + 5 \times 2^3 + 7 \times 2^0$

- Compute $P \ldots 2^3P \ldots 2^8P \ldots 2^{11}P \ldots 2^{14}P \ldots 2^{18}P$
- $d(1)P = 100\,0000\,1000\,0000\,0000$
- $3 \times d(3)P = 000\,0300\,0003\,0000\,0000$
- $5 \times d(5)P = 000\,0000\,0000\,0000\,5000$
- $7 \times d(7)P = 000\,0000\,0000\,0000\,0007$
- $\sum i \times d(i)P = 100\,0300\,1003\,0000\,5007$

$kP = 7d(7)P + 5d(5)P + 3d(3)P + d(1)P$
Same number of operations as the previous methods but slower in practice.

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): sublinear number of non-zero digits

Considered as not suitable for scalar multiplication.

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

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

- More retrictive
- No longer sublinear

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

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

Yao's algorithm adapted to double-base number system

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

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

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

- Same number of doublings/triplings
- Lower number of additions

Generalization

Yao's algorithm can be applied to any number system using two sets of integers

Generalized double-base number system

- $\mathcal{A} = \{a_1, \dots, a_r\}$ and $\mathcal{B} = \{b_1, \dots, b_t\}$ two sets of integers
- $k = \sum_{i=1}^{n} a_{f(i)} b_{g(i)}$ with
  $f : \{1, \dots n\} \rightarrow \{1, \dots r\}$ and
  $g : \{1, \dots n\} \rightarrow \{1, \dots t\}$

Scalar multiplication

- $k = a_1 \sum_{f(i)=1} b_{g(i)} + \cdots + a_n \sum_{f(i)=n} b_{g(i)}$
- Compute the $b_i P$'s for $i = 1 \dots t$
- $d(j)P$ is the sum of all $b_{g(i)} P$'s such that $f(i) = j$
- $kP = a_1 d(1)P + a_2 d(2)P + \cdots + a_n d(n)P$

Examples

Double-base number system

- $\mathcal{A} = \{1, 2, \ldots, 2^{b_{max}}\}$
- $\mathcal{B} = \{1, 3, \ldots, 3^{t_{max}}\}$

Yao's Algorithm

- $\mathcal{A} = \{1, 3, 5 \ldots, 2^w - 1\}$
- $\mathcal{B} = \{1, 2, \ldots, 2^n\}$

Brauer's Algorithm

- $\mathcal{A} = \{1, 2, \ldots, 2^n\}$
- $\mathcal{B} = \{1, 3, 5, \ldots, 2^w - 1\}$

Question

- Can we find better sets?

Binary/Zeckendorf number system

The Fibonacci sequence

- $F_0 = 0, F_1 = 1, \forall n \geq 0, F_{n+2} = F_{n+1} + F_n$
- $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

BZNS

- $\mathcal{A} = \{1, 2, \ldots, 2^{b_{max}}\}$
- $\mathcal{B} = \{F_2, F_3, \ldots, F_{Z_{max}}\}$
- $k = \sum_{i=1}^{n} 2^{b_i} F_{Z_i}$

Computation

Using a greedy approach, just like with the DBNS

Example

$k = 314159 = 2^8 F_{16} + 2^8 F_{13} + 2^5 F_{10} + 2F_9 + 2F_5 + F_2$.
We have $\max(b_i) = 8$ and $\max(Z_i) = 16$:

- Compute $P, 2P, 3P, \ldots, F_{16}P$
- $d(0)P = F_2 P$
- $d(1)P = F_9 P + F_5 P$
- $d(5)P = F_{10} P$
- $d(8)P = F_{16} P + F_{13} P$
- $[k]P = 2^8 d(8)P + 2^5 d(5)P + 2d(1)P + d(0)P$
  $= 2(2^4(2^3 d(8)P + d(5)P) + d(1)P) + d(0)P$

Interesting when the $F_i P$'s can be efficiently computed (like with ECC).

Performing tests

Methodology

For 160-bit scalars and all values of b_{max}, t_{max} and Z_{max} such that $2^{b_{max}}3^{t_{max}}$ and $2^{b_{max}}F_{Z_{max}}$ are 160-bits integers.
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 the DBNS/BZNS systems using the corresponding parameters,
- counted all the operations involved in the point scalar multiplication process.

Curve shape	Method	b_{max}	t_{max} / Z_{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	1260.0
	(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
	Yao-BZNS	142	28	1476.9

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

Remarks

- Yao-DBNS algorithm is less restrictive than double-base chains and faster
- Works with any other double base system
- Gives hope to slow algorithm designers

Future works

- What about multi-base number systems ...
- ... and multi-scalar multiplication ?

Conclusions

Remarks

- Yao-DBNS algorithm is less restrictive than double-base chains and faster
- Works with any other double base system
- Gives hope to slow algorithm designers

Future works

- What about multi-base number systems ...
- ... and multi-scalar multiplication ?

Any questions?

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