On-the-Fly Multi-Base Recoding for ECC Scalar Multiplication without Pre-Computations

Thomas Chabrier and Arnaud Tisserand

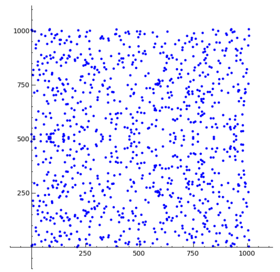
IRISA–CAIRN

ARITH 21, Austin, TX, USA, April 7–10, 2013

Elliptic Curve Cryptography (ECC)

Elliptic curve over $\mathbb{F}_p$:

$$E: \quad y^2 = x^3 + ax + b$$



$y^2 = x^3 + 4x + 20$ over $\mathbb{F}_{1009}$

Curve points representation:

- $P = (x, y)$ affine coordinates ($\mathcal{A}$)
  ☹ many field inversions
- $P = (x, y, z, \ldots)$ redundant coordinates
  ☺ significantly faster
  here we use Jacobian coordinates ($\mathcal{J}$)

Scalar multiplication:

$$Q = [k]P = \underbrace{P + P + \cdots + P}_{k \text{ times}}$$

The most time consuming operation in protocols

where $P \in E$ and $k = (k_{n-1}k_{n-2}\ldots k_1k_0)_2$

k has 160–600 bits

Good and complete presentation in [17] or [7]

Basic Scalar Multiplication

$$Q = [k]P = \underbrace{P + P + \cdots + P}_{k \text{ times}}$$

- $P \in E$
- $k = (k_{n-1}k_{n-2}\ldots k_1k_0)_2$

Double-and-add scalar multiplication algorithm:

1: $Q \leftarrow \mathcal{O}$
2: **for** i **from** $n-1$ **to** 0 **do**
3: $\quad Q \leftarrow [2]Q$ (DBL)
4: $\quad$ **if** $k_i = 1$ **then** $Q \leftarrow Q + P$ (ADD)
5: **return** Q

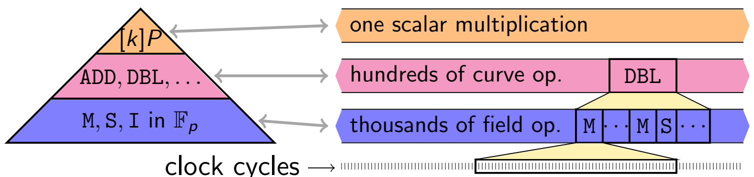
- scans each bit of k and performs corresponding curve-level operation
- average cost: $0.5n$ ADD $+ n$ DBL (security ➜ $\approx 0.5n$ ones in k)
- ADD at line 4 always uses the same point P ➜ keep P in affine and use mADD ($\mathcal{J} + \mathcal{A} \rightarrow \mathcal{J}$)

Curve Level and Field Level Operations

point					
addition	doubling	tripling	quintupling	septupling	...
ADD	DBL	TPL	QPL	SPL	...
$P+Q$ if $P \neq \pm Q$	$[2]P = P+P$	$[3]P = P+P+P$	$[5]P$ $P+\cdots+P$	$[7]P$ $P+\cdots+P$	...

operation at curve level ➜ sequence of ($\approx 10 - -20$) operations at field level

field level op.: add/sub, multiplication: M, square: S, inversion: I



Faster Scalar Multiplication Algorithms

Representation of k impacts #operations → recode k:

- non-adjacent forms (NAF/wNAF):
  high-radix signed-digits representations → increase #0s
- double-base number systems (DBNS): $x = \sum_{i=1}^{n'} d_i b_1^{u_i} b_2^{v_i}$ with $d_i = \pm 1$
  b_1 and b_2 co-prime integers (typically $(b_1, b_2) = (2, 3)$)
  specific op.: point tripling $[3]P = P + P + P$ denoted TPL
  decreasing exponents (Horner form) → higher speed
- multi-base number systems (MBNS):
  more than two bases (co-prime integers), e.g. $(2, 3, 5)$ and $(2, 3, 5, 7)$
  $x = \sum_{i=1}^{n'} \left(d_i \prod_{j=1}^{l} b_j^{e_{j,i}} \right)$ with $d_i = \pm 1$

BUT those recoding methods require pre-computations:

- wNAF: pre-compute and store $P_j = [j]P \quad \forall j \in \{3, 5, \ldots, 2^{w-1} - 1\}$
- DBNS/MBNS recoding is performed off-line

Remark: point subtraction (SUB) is as efficient as point addition

Our Goals

MBNS recoding and ECC scalar multiplication:

- fully implemented in FPGA (ASIC version is underway)
- without pre-computations
- recoding is performed in parallel to curve-level operations
- fine tuning of architecture parameters
- presented for curves defined over $\mathbb{F}_p$ (due to space limit)
  also works for $\mathbb{F}_{2^m}$ with slightly different fine tuning

Notations

- $k = (k_{n-1} k_{n-2} \ldots k_1 k_0)_2$, $k > 1$, the n-bit scalar stored into t words of w bits with $w(t-1) < n \leq wt$ (i.e. last word may be 0-padded).
  $k^{(i)}$ the ith word of k starting from least significant for $0 \leq i < t$
- $\mathcal{B}$ the multi-base with l base elements (co-prime integers),
  $\mathcal{B} = (b_1, b_2, \ldots, b_l)$
- predicate $\mathrm{divisible}(x, \mathcal{B})$ returns true if x is divisible by at least one base element in $\mathcal{B}$ (false for other cases)
- number x represented as the sum of terms $x = \sum_{i=1}^{n'} \left(d_i \prod_{j=1}^{l} b_j^{e_{j,i}} \right)$
  with $d_i \in \{0, \pm 1\}$ and in Horner form
- term $(d_i, e_{1,i}, e_{2,i}, \ldots, e_{l,i})$ defined by $d_i \times \prod_{j=1}^{l} b_j^{e_{j,i}}$ in $\mathcal{B}$ (index i may be omitted when context is clear)
- Q, P curve points and $Q = [k]P$ scalar multiplication

Very Simple MBNS Unsigned Recoding Algorithm

Transforms k into a list of terms (LT) in Horner form

```
 1: LT ← ∅
 2: while k > 1 do
 3:     if not( divisible(k, B) ) then                (divisibility test)
 4:         d ← 1
 5:         k ← k − 1
 6:     else
 7:         d ← 0
 8:     for j from 1 to l do
 9:         e_j ← 0
10:         while k ≡ 0 mod b_j do                    (divisibility test)
11:             e_j ← e_j + 1
12:             k ← k / b_j                           (exact division)
13:     LT ← LT ∪ (d, e_1, e_2, ..., e_l)
14: return LT
```

Remark: divisibility tests at line 3 and 10 are shared

Very Simple MBNS Scalar Multiplication Algorithm

- MBNS recoding works in a serial way starting with most significant
- each term can be immediately used in the scalar multiplication
  $\longrightarrow$ recorded terms are processed and used on-the-fly
- multi-base adaptation of standard left-to-right scalar multiplication ([17, Sec. 3.3.1])

$Q \leftarrow \mathcal{O}$
foreach t **in** LT **do** $\qquad (t = (d, e_1, e_2, \ldots, e_l))$
  $Q \leftarrow Q + d \times P$ $\qquad (d \in \{0, 1\} \Rightarrow$ NOP/ADD$)$
  for j **from** 1 **to** l **do**
    $P \leftarrow [b_j^{e_j}]P$ $\qquad$ (DBL, TPL, QPL, $\ldots$)
  $Q \leftarrow Q + P$
return Q

Remark 1: recoding and curve-level operations are overlapped
Remark 2: P is modified over time, we cannot use mADD (time penalty)
Remark 3: $d = 0$ is only possible for the very first term

Implementation of Divisibility Tests (1/2)

We use Pascal's tapes, [28] (published in 1819), [31], values are $2^i \bmod b_j$:

b_j \ i	11	10	9	8	7	6	5	4	3	2	1	0
3	2	1	2	1	2	1	2	1	2	1	**2**	**1**
5	3	4	2	1	3	4	2	1	**3**	**4**	**2**	**1**
7	4	2	1	4	2	1	4	2	1	**4**	**2**	**1**

For $b_j = 3$, the periodic sequence is $(2\,1)^*$:

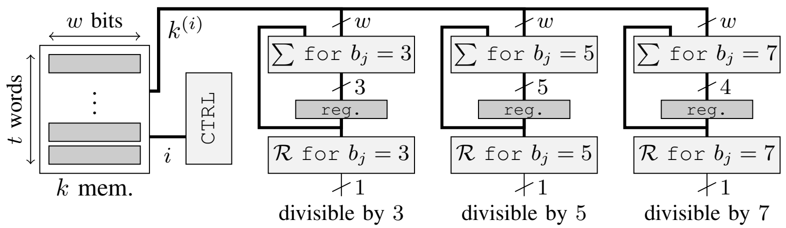
$$\begin{aligned} k \bmod 3 &= (\ldots + 2^3 k_3 + 2^2 k_2 + 2^1 k_1 + k_0) \bmod 3 \\ &= (\ldots + 2k_3 + k_2 + 2k_1 + k_0) \bmod 3 \\ &= \Big(\underbrace{\sum (2k_{2i+1} + k_{2i})}_{\alpha} \Big) \bmod 3. \end{aligned}$$

For $b_j = 5$, the periodic sequence is $(3\,4\,2\,1)^*$

- use $3 = 1 + 2$ $\longrightarrow$ unsigned sum with additional inputs (FPGA)
- use $3 \equiv -2 \bmod 5$ $\longrightarrow$ signed sum with less operands

Implementation of Divisibility Tests (2/2)

To avoid complex decoding, we use $w = \mathrm{lcm}(2, 4, 3) = 12$ and $w = 24$



FPGA results for $n = 160$ (XC5VLX50T, ISE 12.4, std efforts S/P&R):

w	area slices (FF/LUT)	freq. MHz	clock cycles
12	25 (40/81)	543	$t + 3$
24	67 (53/152)	549	$t + 4$

Implementation of Exact Division by b_j Elements (1/2)

Exact division k/b_j: we know that k is divisible by b_j

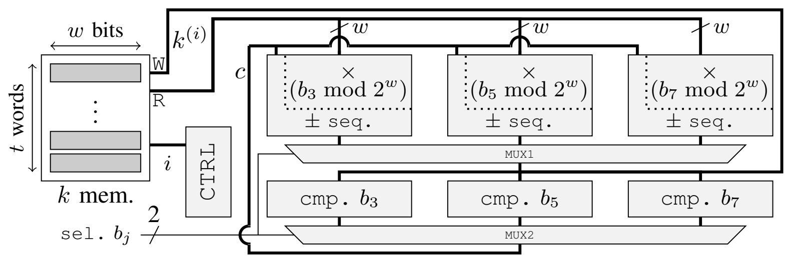
Algorithm from [19] (LSWF), optimized for FPGA and $b_j \in \{3, 5, 7\}$:

$c \leftarrow 0$
for i **from** 0 **to** $t - 1$ **do**
  $r \leftarrow k^{(i)} - c$
  $r \leftarrow r \times (b_j^{-1} \bmod 2^w)$
  $c \leftarrow 0$
  for h **from** 1 **to** $b_j - 1$ **do**
    if $r \geq h \times \lceil (2^w - 1)/b_j \rceil$ **then**
      $c \leftarrow c + 1$
  $k^{(i)} \leftarrow (r_{w-1} \cdots r_0)$
return k

b_j	$b_j^{-1} \bmod 2^{12}$, γ	$b_j^{-1} \bmod 2^{24}$, γ
3	$(101010101011)_2$, 3	$(101010101010101010101011)_2$, 4
5	$(110011001101)_2$, 3	$(110011001100110011001101)_2$, 4
7	$(110110110111)_2$, 3	$(110110110110110110110111)_2$, 4

We use our multiplication by constant algorithm [4]

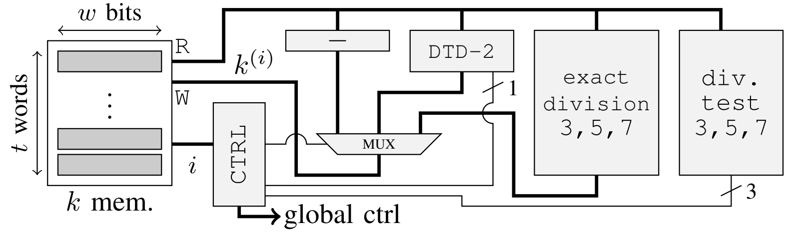
Implementation of Exact Division by b_j Elements (2/2)



FPGA results for $n = 160$ (XC5VLX50T, ISE 12.4, std efforts S/P&R):

w	area slices (FF/LUT)	freq. MHz	clock cycles
12	59 (138/171)	291	$t+4$
24	152 (441/448)	202	$t+5$

Unsigned Multiple-Base Recoding Unit



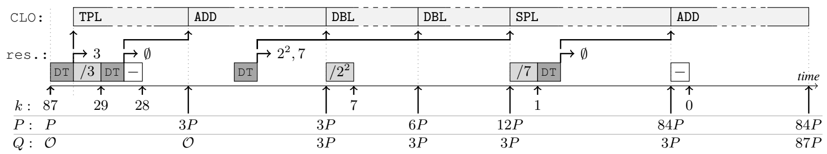
FPGA results for $n = 160$ and $\mathcal{B} = (2, 3, 5, 7)$ (XC5VLX50T, ISE 12.4, std efforts S/P&R):

w	area slices (FF/LUT)	freq. MHz
12	153 (301/412)	232
24	323 (682/908)	202

Remark: DTD-2 divisibility test and division by $2^{1\ldots v}$ with $v \leq w$

Example

$$87 = 0 + 3^1 \times (1 + 2^2 7^1)$$



Notations:

- "CLO" denotes curve-level operations
- DT denotes divisibility test, "res." their results
- "$/b_j$" exact division by b_j

Remark: very short latency at the very beginning ($< 0.01\,\%$ of $[k]P$ for $n = 160$ and even less for larger fields)

Signed Digits Version: $d \in \{0, \pm 1\}$

Add a selection function S in the recoding algorithm:

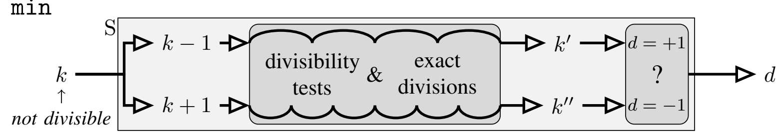
unsigned version	
4:	$d \leftarrow 1$
5:	$k \leftarrow k - 1$

$\longrightarrow$

signed version	
4:	$d \leftarrow \mathrm{S}(k)$
5:	$k \leftarrow k - d$

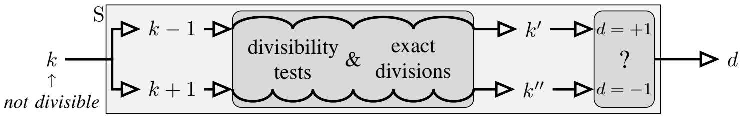
We compared 4 heuristic selection functions:

- min



- approx: approximated minimum value selection function
- max_nb_div: maximum number of divisors selection function
- min2: 2 steps minimum value selection function
  1) $(k-1, k+1) \xrightarrow{\texttt{min}} (k', k'')$
  2) $(k'-1, k'+1, k''-1, k''+1) \xrightarrow{\texttt{min2}} (\zeta', \zeta'', \zeta''', \zeta'''')$

`approx` Selection Function



Computing (k', k'') is expensive, so we try to get an approximation

$$k' \approx \delta' = \underbrace{\lfloor \log_2(k-1) \rfloor + 1}_{\text{MSB position of } k-1} - \sum_{j=1}^{l} e'_j \log_2(b_j)$$

$$k'' \approx \delta'' = \underbrace{\lfloor \log_2(k+1) \rfloor + 1}_{\text{MSB position of } k+1} - \sum_{j=1}^{l} e''_j \log_2(b_j)$$

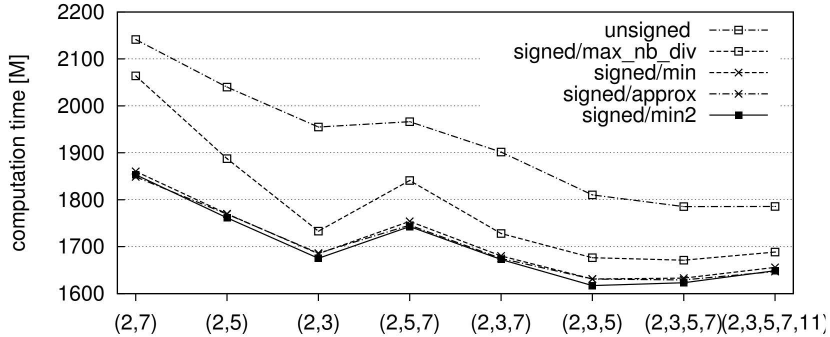
1) Exponents e'_j and e''_j are produced by the divisibility tests

2) Approximate constants: $\log_2 3 \approx 1.59$, $\log_2 5 \approx 2.32$, and $\log_2 7 \approx 2.81$

$$\delta' = \mathrm{MSB(k-1)} - e_{b_1=2} - 1.5 e_{b_2=3} - 2.25 e_{b_3=5} - 2.75 e_{b_4=7}$$

Comparison of Selection Functions

For curves over $\mathbb{F}_p$ with $a = -3$:



Average computation time (in M) of 10 000 scalar multiplications with 160-bit values

Similar behavior for curves with $a \neq -3$

Complete FPGA Implementation Results

Signed recoding unit with `approx` heuristic:

w	area slices (FF/LUT)	freq. MHz
12	173 (326/466)	232
24	345 (724/1 005)	202

ECC processor (modification from [6], collab. UCC crypto group):

version	memory type	area slices (FF/LUT)	area BRAM	freq. MHz
small	distributed	2 204 (3 971/6 816)	0	155
	BRAM	1 793 (3 641/6 182)	6	155
large	distributed	3 182 (4 668/7 361)	0	142
	BRAM	2 427 (4 297/6 981)	6	142

- small: $\mathbb{F}_p$ curves, $n = 160$, Jacob. coord., NAF/MBNS, 1 unit/op.
- large: same with 4NAF/MBNS and 2 $\pm$, 2 $\times$, 1 inv.

$[k]P$ Timings Comparison

For $n = 160$ and $a \neq -3$:

refs.	methods	perfs	pre-computations storage	pre-computations operations	recoding
	dbl&add	1 985.3M	∅	∅	∅
	NAF	1 723.0M	∅	∅	on-the-fly & very cheap
	3NAF	1 583.7M	1 pt	49.4M	on-the-fly & very cheap
	4NAF	1 499.1M	3 pts	140.8M	on-the-fly & very cheap
[10]	DBNS	1 863.0M	∅	∅	off-line & costly
[11]	DBNS	1 722.3M	∅	∅	off-line & costly
[3]	DBNS	1 558.4M	7 pts	>150M	off-line & costly
[15]	DBNS	1 615.3M	∅	∅	off-line & costly
our	(2, 3) MBNS	1 746.2M	∅	∅	on-the-fly & small
	(2, 3, 5) MBNS	1 679.9M	∅	∅	on-the-fly & small
	(2, 3, 5, 7) MBNS	1 670.4M	∅	∅	on-the-fly & small

For $n = 160$ and $a = -3$: about 15% slower than best DBNS/MBNS (theoretical) solutions

Conclusion & Future Prospects

- first full hardware implementation of MBNS recoding and ECC scalar multiplication
- even a simple MBNS recoding is:
  - not so slow $\approx +15\%$ compared the fastest solutions
  - not so big $\approx +10\%$ on Virtex 5 FPGAs

Future works:

- ASIC version (underway)
- advanced recodings for higher speed and better protection against SCAs

Acknowledgments:

- Prof. Christiane Frougny for historical references
- Anonymous reviewers for their valuable comments
- *Région Bretagne* / *Conseil Général des Côtes d'Armor* (ROBUSTA prj): PhD grant
- PAVOIS project (ANR 12 BS02 002 01): partial funding

The end, some questions ?

Contact:

- `mailto:arnaud.tisserand@irisa.fr`
- `http://people.irisa.fr/Arnaud.Tisserand/`
- CAIRN Group `http://www.irisa.fr/cairn/`
- IRISA Laboratory, CNRS–INRIA–Univ. Rennes 1
  6 rue Kerampont, CS 80518, F-22305 Lannion cedex, France

Thank you

ANR PAVOIS `http://pavois.irisa.fr/`

- ANR Project ANR 2012–2016
- IRISA (Lannion) + LIRMM (Perpignan & Montpellier)
- Arithmetic Protections Against Physical Attacks for Elliptic Curve based Cryptography

References by Topics

- ECC: [17] and [7]
- DBNS: [9], [12], [13], [10], [15], [2], [3], [14] and [11]
- MBNS: [26], [21], [23], [29], [32], [1] and [30]
- Side-channel attacks and counter-measures: [25], [27] [20], [8], [5]
- Pascal's tape: [28], [31]
- Exact division: [19]
- Multiplication by constant: [4]
- ...

References I

[1] J. Adikari, V. S. Dimitrov, and L. Imbert.
Hybrid binary-ternary number system for elliptic curve cryptosystems.
IEEE Transactions on Computers, 60(2):254–265, February 2011.

[2] R. Barua, S. K. Pandey, and R. Pankaj.
Efficient window-based scalar multiplication on elliptic curves using double-base number system.
In *Proc. 8th International Conference on Progress in Cryptology (INDOCRYPT)*, volume 4859 of *LNCS*, pages 351–360. Springer, December 2007.

[3] D. J. Bernstein, P. Birkner, T. Lange, and C. Peters.
Optimizing double-base elliptic-curve single-scalar multiplication.
In *Proc. 8th International Conference on Progress in Cryptology (INDOCRYPT)*, volume 4859 of *LNCS*, pages 167–182. Springer, December 2007.

[4] N. Boullis and A. Tisserand.
Some optimizations of hardware multiplication by constant matrices.
IEEE Transactions on Computers, 54(10):1271–1282, October 2005.

[5] A. Byrne, N. Meloni, A. Tisserand, E. M. Popovici, and W. P. Marnane.
Comparison of simple power analysis attack resistant algorithms for an elliptic curve cryptosystem.
Journal of Computers, 2(10):52–62, 2007.

[6] A. Byrne, E. Popovici, and W.P. Marnane.
Versatile processor for gf(p^m) arithmetic for use in cryptographic applications.
IET Computers & Digital Techniques, 2(4):253–264, July 2008.

[7] H. Cohen and G. Frey, editors.
Handbook of Elliptic and Hyperelliptic Curve Cryptography.
Chapman & Hall/CRC, 2005.

[8] J.-S. Coron.
Resistance against differential power analysis for elliptic curve cryptosystems.
In *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, volume 1717 of *LNCS*, pages 292–302. Springer, August 1999.

References II

[9] V. Dimitrov and T. Cooklev.
Hybrid algorithm for the computation of the matrix polynomial $I + A + \ldots + A^{N-1}$.
IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 42(7):377–380, July 1995.

[10] V. Dimitrov, L. Imbert, and P. K. Mishra.
Efficient and secure elliptic curve point multiplication using double-base chains.
In *Proc. 11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, volume 3788 of *LNCS*, pages 59–78. Springer, December 2005.

[11] V. Dimitrov, L. Imbert, and P. K. Mishra.
The double-base number system and its application to elliptic curve cryptography.
Mathematics of Computation, 77(262):1075–1104, April 2008.

[12] V.S. Dimitrov, G.A. Jullien, and W.C. Miller.
An algorithm for modular exponentiation.
Information Processing Letters, 66(3):155–159, May 1998.

[13] V.S. Dimitrov, G.A. Jullien, and W.C. Miller.
Theory and applications of the double-base number system.
IEEE Trans. on Computers, 48(10):1098–1106, October 1999.

[14] C. Doche and L. Habsieger.
A tree-based approach for computing double-base chains.
In *Proc. 13th Australasian Conference on Information Security and Privacy*, volume 5107 of *LNCS*, pages 433–446, July 2008.

[15] C. Doche and L. Imbert.
Extended double-base number system with applications to elliptic curve cryptography.
In *Proc. 7th International Conference on Cryptology (INDOCRYPT)*, volume 4329 of *LNCS*, pages 335–348, Kolkata, India, December 2006. Springer.

[16] P. Giorgi, L. Imbert, and T. Izard.
Optimizing elliptic curve scalar multiplication for small scalars.
In *Proc. Mathematics for Signal and Information Processing*, volume 7444, pages 74440N:1–10. SPIE, September 2009.

References III

[17] D. Hankerson, A. Menezes, and S. Vanstone.
Guide to Elliptic Curve Cryptography.
Springer, 2004.

[18] K. Itoh, M. Takenaka, N. Torii, S. Temma, and Y. Kurihara.
Fast implementation of public-key cryptography on a DSP TMS320C6201.
In *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, volume 1717 of *LNCS*, pages 61–72. Springer, August 1999.

[19] T. Jebelean.
An algorithm for exact division.
Journal of Symbolic Computation, 15(2):169–180, February 1993.

[20] M. Joye.
Advances in Elliptic Curve Cryptography, volume 317 of *London Mathematical Society Lecture Note*, chapter Defenses Against Side-Channel Analysis, pages 87–100.
Cambridge University Press, April 2005.

[21] P. Longa.
Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields.
Master's thesis, Univ. Ottawa, 2007.

[22] P. Longa and C. Gebotys.
Setting speed records with the (fractional) multibase non-adjacent form method for efficient elliptic curve scalar multiplication.
Technical Report 118, Cryptology ePrint Archive, 2008.

[23] P. Longa and C. Gebotys.
Fast multibase methods and other several optimizations for elliptic curve scalar multiplication.
In *Proc. Public Key Cryptography (PKC)*, volume 5443 of *LNCS*, pages 443–462, 2009.

[24] P. Longa and A. Miri.
New multibase non-adjacent form scalar multiplication and its application to elliptic curve cryptosystems.
Technical Report 52, IACR Eprint, 2008.

References IV

[25] S. Mangard, E. Oswald, and T. Popp.
Power Analysis Attacks: Revealing the Secrets of Smart Cards.
Springer, 2007.

[26] P. K. Mishra and V. Dimitrov.
Efficient quintuple formulas for elliptic curves and efficient scalar multiplication using multibase number representation.
In *Proc. 10th International Conference on Information Security (ISC)*, volume 4779 of *LNCS*, pages 390–406, October 2007.

[27] E. Oswald.
Advances in Elliptic Curve Cryptography, volume 317 of *London Mathematical Society Lecture Note Series*, chapter Side Channel Analysis, pages 69–86.
Cambridge University Press, April 2005.

[28] B. Pascal.
Œuvres complètes, volume 5, chapter De Numeribus Multiplicibus, pages 117–128.
Librarie Lefèvre, 1819.

[29] G. N. Purohit and A. S. Rawat.
Fast scalar multiplication in ECC using the multi base number system.
International Journal of Computer Science Issues, 8(1):131–137, May 2011.

[30] G.N. Purohit, A. S. Rawat, and M. Kumar.
Elliptic curve point multiplication using MBNR and point halving.
International Journal of Advanced Networking and Applications, 3(5):1329–1337, 2012.

[31] J. Sakarovitch.
Elements of Automata Theory, chapter Prologue: M. Pascal's Division Machine, pages 1–6.
Cambridge, 2009.

[32] X. Yin, T. Yang, and J. Ning.
Optimized approach for computing multi-base chains.
In *Proc. 7th International Conference on Computational Intelligence and Security (CIS)*, pages 964–968. IEEE, December 2011.

Backup Slides

Costs of Curve Level Operations

Best computation costs from literature and curves over $\mathbb{F}_p$

a		curve-level operations					
-3	refs.	ADD	mADD	DBL	TPL	QPL	SPL
$\neq$	EFD	$11\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+4\mathtt{S}$	$1\mathtt{M}+8\mathtt{S}$	$5\mathtt{M}+10\mathtt{S}$	n. a.	n. a.
	[16]	n. a.	n. a.	$1\mathtt{M}+8\mathtt{S}$	$5\mathtt{M}+10\mathtt{S}$	$7\mathtt{M}+16\mathtt{S}$	$15\mathtt{M}+24\mathtt{S}$
	[22]	$11\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+4\mathtt{S}$	$2\mathtt{M}+8\mathtt{S}$	$6\mathtt{M}+11\mathtt{S}$	$9\mathtt{M}+15\mathtt{S}$	$13\mathtt{M}+18\mathtt{S}$
$=$	EFD	$11\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+4\mathtt{S}$	$3\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+7\mathtt{S}$	n. a.	n. a.
	[24]	$11\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+4\mathtt{S}$	$3\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+7\mathtt{S}$	$11\mathtt{M}+11\mathtt{S}$	$18\mathtt{M}+11\mathtt{S}$
	[23][22]	$11\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+4\mathtt{S}$	$3\mathtt{M}+5\mathtt{S}$	$7\mathtt{M}+8\mathtt{S}$	$10\mathtt{M}+12\mathtt{S}$	$14\mathtt{M}+15\mathtt{S}$

	refs.	λDBL	λTPL
$\neq$	[10][11][18]	$4\lambda\mathtt{M}+(4\lambda+2)\mathtt{S}$	$(11\lambda-1)\mathtt{M}+(4\lambda+2)\mathtt{S}$

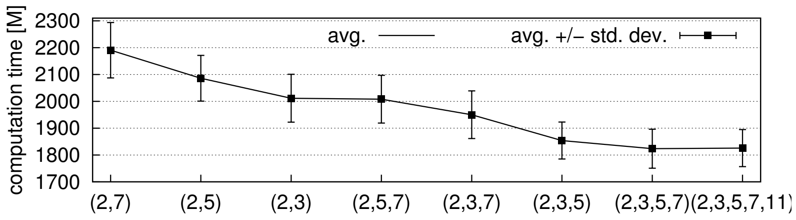
	refs.	λTPL / λ'DBL
$\neq$	[10][11]	$(11\lambda+4\lambda'-1)\mathtt{M}+(4\lambda+4\lambda'+3)\mathtt{S}$

EFD: Explicit-Formulas Database `http://hyperelliptic.org/EFD`

$\mathtt{mADD}: \mathcal{A}+\mathcal{J} \longrightarrow \mathcal{J}$

Statistical Timings of Unsigned MBNS Scalar Multiplication

Goal: selection of $\mathcal{B}$ elements



- complete $[k]P$ timings (in M) for 10 000 random 160-bit values
- most efficient multi-base is $\mathcal{B}=(2,3,5,7)$
- adding $b_j=11$ does not improve the performance while it makes the architecture larger and slower
- $b_1=2$ for all configurations (k is received in binary)

Randomized Selection Function

When k is not divisible by $\mathcal{B}$ elements, S returns $d=1$ or $d=-1$ randomly as a simple protection against some side-channel attacks

Average computation times for $a \neq -3$ and 10 000 random scalars:

$\mathcal{B}$	rnd		min		diff.
	M	#ADD	M	#ADD	[%]
(2,3)	1 960.5	49.3	1 738.5	34.0	12.8
(2,3,5)	1 843.0	39.8	1 673.7	28.0	10.1
(2,3,5,7)	1 811.4	34.8	1 670.0	24.8	8.5
(2,3,5,7,11)	1 816.7	32.1	1 693.5	22.9	7.3

DBNS and MBNS are very redundant and sparse representations

Security efficiency has to be evaluated

Comparisons for $n=160$ and $a=-3$

references	methods	performances	pre-computations storage	pre-computations operations	recoding
	double-and-add	1 922.0M	$\emptyset$	$\emptyset$	$\emptyset$
	NAF	1 659.7M	$\emptyset$	$\emptyset$	on-the-fly & very cheap
	3NAF	1 520.2M	1 point	49.0M	on-the-fly & very cheap
	4NAF	1 436.1M	3 points	140.0M	on-the-fly & very cheap
[15]	DBNS	1 563.2M	$\emptyset$	$\emptyset$	off-line & costly
[3]	DBNS	1 504.3M	7 points	$>$150M	off-line & costly
[26]	(2, 3, 5)MBNS	1 645.4M	$\emptyset$	$\emptyset$	off-line & costly
		1 606.4M	1 point	$\approx$45M	off-line & costly
		1 566.4M	3 points	$\approx$150M	off-line & costly
		1 552.3M	7 points	$>$150M	off-line & costly
		1 486.4M	5 points	$>$150M	off-line & costly
[24]	(2, 3)NAF	1 514.0M	$\emptyset$	$\emptyset$	small
	(2, 3, 5)NAF	1 490.0M	$\emptyset$	$\emptyset$	small
	(2, 3, 5, 7)NAF	1 491.0M	$\emptyset$	$\emptyset$	small
	(2, 3)NAF_3	1 460.0M	1 point	$\approx$45M	small
	(2, 3, 5)NAF_3	1 444.0M	1 point	$\approx$45M	small
	(2, 3, 5, 7)NAF_3	1 449.0M	1 point	$\approx$45M	small
	(2, 3)NAF_4	1 384.0M	3 points	$>$150M	small
	(2, 3, 5)NAF_4	1 383.0M	3 points	$>$150M	small
	(2, 3, 5, 7)NAF_4	1 394.0M	3 points	$>$150M	small
[23]	(2, 3, 5)NAF	1 460.0M	$\emptyset$	$\emptyset$	costly
	(2, 3, 5)NAF	1 426.0M	6 points	$>$150M	costly
this work	(2, 3)MBNS	1 686.2M	$\emptyset$	$\emptyset$	on-the-fly & small
	(2, 3, 5)MBNS	1 631.0M	$\emptyset$	$\emptyset$	on-the-fly & small
	(2, 3, 5, 7)MBNS	1 629.3M	$\emptyset$	$\emptyset$	on-the-fly & small