AsiaCrypt 2014 – Kaohsiung
9th December 2014

On the Enumeration of DBCs with Applications to ECC

Christophe Doche

Macquarie University

Sydney – Australia

Main Contributions

First algorithm to find optimal DBCs

Relies on the enumeration of all the chains representing a given integer

New concept of Controlled DBC

Create a random chain from scratch

Enumerate all the chains with given properties to select the parameters

Scalar Multiplication

Definition. Given an integer n and a point P on a curve, a scalar multiplication consists in computing

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

It is the core operation in most ECC protocols

Double-and-Add Method

The standard way to compute $[n]P$ is the double-and-add method

The method uses the following operations:

- addition $P + Q$, when $P \neq \pm - Q$
- doubling $[2]P$

It relies on the binary representation of n

Double-Base Number System

Represent the scalar n as

$$n = \sum_{i=1}^{\ell} c_i 2^{a_i} 3^{b_i}, \quad \text{with} \quad c_i = \pm 1$$

Such an expansion can be easily found with a greedy approach

Double-Base Number System

Example.

We have

$$841232 = 2^{10}3^6 + 2^7 3^6 + 2^1 3^6 - 2^2 3^2 + 2$$

DBNS expansions are in general not very well suited to compute scalar multiplications

Double-Base Chain

Represent the scalar n as

$$n = \sum_{i=1}^{\ell} c_i 2^{a_i} 3^{b_i}, \quad \text{with} \quad c_i = \pm 1$$

with

$$a_1 \geqslant a_2 \geqslant \ldots \geqslant a_\ell \quad \text{and} \quad b_1 \geqslant b_2 \geqslant \ldots \geqslant b_\ell$$

This simple contraint makes the computation of $[n]P$ a lot more straightforward

Double-Base Chain

Example.

We have

$$841232 = 2^{10}3^6 + 2^7 3^6 + 2^1 3^6 - 3^3 - 3^2 + 3 - 1$$

From which we compute

Double-Base Chain

Example.

We have

$$841232 = 2^{10}3^6 + 2^7 3^6 + 2^1 3^6 - 3^3 - 3^2 + 3 - 1$$

From which we compute

$$[2^3]P$$

Double-Base Chain

Example.

We have

$$841232 = 2^{10}3^6 + 2^{7}3^6 + 2^{1}3^6 - 3^3 - 3^2 + 3 - 1$$

From which we compute

$$[2^{6}]([2^3]P + P)$$

Double-Base Chain

Example.

We have

$$841232 = 2^{10}3^6 + 2^7 3^6 + 2^1 3^6 - 3^3 - 3^2 + 3 - 1$$

From which we compute

$$[2^1 3^3]\Big([2^6]([2^3]P + P) + P\Big)$$

Double-Base Chain

Example.

We have

$$841232 = 2^{10}3^6 + 2^7 3^6 + 2^1 3^6 - 3^3 - 3^2 + 3 - 1$$

From which we compute

$$[3^1]\Big([2^1 3^3]\Big([2^6]([2^3]P + P) + P\Big) - P\Big)$$

and so on

Double-Base Chain

Can we do better?

Double-Base Chain

Can we do better?

This question needs to be refined

Indeed, assume that

$$841232 = 2^{a_1}3^{b_1} - 2^{a_2}3^{b_2}$$

for very large a_1, b_1, a_2, b_2

That is not going to help computing $[841232]P$ efficiently

Double-Base Chain

Terminology.

Consider the DBC

$$n = 2^{a_1}3^{b_1} + c_2 2^{a_2}3^{b_2} + \cdots + c_\ell 2^{a_\ell}3^{b_\ell}$$

Then

$2^{a_1}3^{b_1}$ is the leading factor

ℓ is the length

Double-Base Chain

The leading factor and the length of a DBC fully capture the complexity of computing $[n]P$ with this DBC

We need a_1 doublings, b_1 triplings and $\ell - 1$ additions to compute $[n]P$

Double-Base Chain

Definition. Take a scalar n and two integers a and b

A DBC with a leading factor dividing $2^a 3^b$ is optimal for n if its length ℓ is minimal across all the DBCs representing n and having a leading factor dividing $2^a 3^b$

A Partition Problem

In 1979, Erdős and Loxton study the number $p(n)$ of partitions of n of the form

$$n = d_k + \cdots + d_2 + d_1 \quad \text{with} \quad d_1 \mid d_2 \mid \cdots \mid d_k$$

A Partition Problem

In 1979, Erdős and Loxton study the number $p(n)$ of partitions of n of the form

$$n = d_k + \cdots + d_2 + d_1 \quad \text{with} \quad d_1 \mid d_2 \mid \cdots \mid d_k$$

For that, they introduce $p_1(n)$ as the number of partitions of n of the form

$$n = d_k + \cdots + d_2 + 1 \quad \text{with} \quad d_2 \mid \cdots \mid d_k$$

A Partition Problem

They observe that

$$p(n) = p_1(n) + p_1(n+1)$$

A Partition Problem

They observe that

$$p(n) = p_1(n) + p_1(n+1)$$

and that

$$p_1(n) = \sum_{\substack{d \mid n-1 \\ d>1}} p_1\left(\frac{n-1}{d}\right)$$

Another Partition Problem

Let $q(a, b, n)$ the number of signed partitions of n of the form

$$n = d_k \pm d_{k-1} \pm \cdots \pm d_2 \pm d_1$$

$$\text{with} \;\; d_1 \mid d_2 \mid \cdots \mid d_k \mid 2^a 3^b$$

It is clear that $q(a, b, n)$ is the number of DBCs with a leading factor dividing $2^a 3^b$ and representing n

Another Partition Problem

We also introduce $q_1(a, b, n)$ for

$$n = d_k \pm d_{k-1} \pm \cdots \pm d_2 + 1$$

$$\text{with} \quad d_2 \mid \cdots \mid d_k \mid 2^a 3^b$$

and $q_{\bar{1}}(a, b, n)$ for

$$n = d_k \pm d_{k-1} \pm \cdots \pm d_2 - 1$$

$$\text{with} \quad d_2 \mid \cdots \mid d_k \mid 2^a 3^b$$

Another Partition Problem

We have

$$q(a,b,n) = q_1(a,b,n) + q_{\bar{1}}(a,b,n) + q_{\bar{1}}(a,b,n+1)$$

$$\begin{aligned} q_1(a,b,n) = & \sum_{\substack{d \mid \gcd(n-1, 2^a 3^b) \\ d>1}} q_1\left(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \frac{n-1}{d}\right) \\ & + \sum_{\substack{d \mid \gcd(n-1, 2^a 3^b) \\ d>1}} q_{\bar{1}}\left(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \frac{n-1}{d}\right) \end{aligned}$$

Another Partition Problem

We have

$$q(a,b,n) = q_1(a,b,n) + q_{\bar{1}}(a,b,n) + q_{\bar{1}}(a,b,n+1)$$

$$\begin{aligned} q_{\bar{1}}(a,b,n) = & \sum_{\substack{d \mid \gcd(n+1, 2^a 3^b) \\ d>1}} q_1\left(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \frac{n+1}{d}\right) \\ & + \sum_{\substack{d \mid \gcd(n+1, 2^a 3^b) \\ d>1}} q_{\bar{1}}\left(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \frac{n+1}{d}\right) \end{aligned}$$

Another Partition Problem

We deduce a recursive algorithm to compute $q(a, b, n)$

Another Partition Problem

We deduce a recursive algorithm to compute $q(a, b, n)$

Furthermore, a simple modification allows to keep track of the length of the DBCs

Let $q(a, b, \ell, n)$ be the number of signed partitions of n of the form

$$n = d_k \pm d_{k-1} \pm \cdots \pm d_2 \pm d_1$$

$$\text{with} \quad d_1 \mid d_2 \mid \cdots \mid d_k \mid 2^a 3^b \quad \text{and} \quad k \leqslant \ell$$

Another Partition Problem

We have

$$\begin{aligned} q(a,b,\ell,n) &= q_1(a,b,\ell,n) \\ &+ q_{\bar{1}}(a,b,\ell,n) \\ &+ q_{\bar{1}}(a,b,\ell+1,n+1) \end{aligned}$$

Additionally, $q_1(a,b,\ell,n)$ and $q_{\bar{1}}(a,b,\ell,n)$ satisfy similar relations than previously

Another Partition Problem

Algorithm. $q_1(a, b, \ell, n)$

Input: An integer n and parameters a, b, and ℓ.
Output: Number of DBCs ending with 1 with a leading factor dividing $2^a 3^b$, and length less than or equal to ℓ.

if $n \leqslant 0$ **or** $a < 0$ **or** $b < 0$ **or** $\ell \leqslant 0$ **then return** 0
else if $n = 1$ **then**
    if $a \geqslant 0$ **and** $b \geqslant 0$ **then return** $\min(1, \max(0, \ell))$
    else return 0
else if $n > 1$ **then**
    $D \leftarrow \gcd(n-1, 2^a 3^b)$
    $s \leftarrow 0$
    for each divisor $d > 1$ of D **do**
        $s \leftarrow s + q_1\left(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \ell - 1, \frac{n-1}{d}\right)$
        $s \leftarrow s + q_{\bar{1}}\left(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \ell - 1, \frac{n-1}{d}\right)$
return s

Another Partition Problem

Algorithm. $q_{\bar{1}}(a, b, \ell, n)$

INPUT: An integer n and parameters a, b, and ℓ.
OUTPUT: Number of DBCs ending with -1 with a leading factor dividing $2^a 3^b$, and a length less than or equal to ℓ.

if $n \leqslant 0$ **or** $a < 0$ **or** $b < 0$ **or** $\ell \leqslant 0$ **then return** 0
else if $n = 1$ **then**
    if $a \geqslant 0$ **and** $b \geqslant 0$ **then return** $\min\big(a, \max(0, \ell - 1)\big)$
    else return 0
else if $n > 1$ **then**
    $D \leftarrow \gcd(n + 1, 2^a 3^b)$
    $s \leftarrow 0$
    for each divisor $d > 1$ of D **do**
        $s \leftarrow s + q_1\big(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \ell - 1, \frac{n+1}{d}\big)$
        $s \leftarrow s + q_{\bar{1}}\big(a - \mathrm{val}_2(d), b - \mathrm{val}_3(d), \ell - 1, \frac{n+1}{d}\big)$
return s

Optimal DBC

Given n, a and b, we can then compute $q(a, b, \ell, n)$ for increasing values of ℓ

This gives the length of an optimal DBC for n

Another straightforward modification in the algorithm allows to actually return an optimal DBC

Optimal DBC

Example. We find that $q(12, 6, 4, 841232) = 0$ and $q(12, 6, 5, 841232) = 3$

In other words, using at most 12 doublings and 6 triplings, the shortest chain to compute $[841232]P$ is of length 5

The algorithm returns

$$841232 = 2^{10}3^6 + 2^7 3^6 + 2^4 3^4 + 2^4 3^2 - 2^4$$

Optimal DBC

In general, it is quite fast to find optimal chains of length up to 12

It took several hours to find an optimal chain of length 18 corresponding to a random integer of size 69 bits

It is not realistic to expect finding an optimal DBC for a scalar of say size 200 bits

Controlled DBC

Instead of generating a random integer n and then trying to find a short DBC to represent it

Select a leading factor $2^a 3^b$ and a length ℓ then generate a random DBC from scratch

The question then becomes:

What value of ℓ is long enough?

Controlled DBC

Fix a, b, ℓ

1. Determine the interval of integers that can be represented with those chains

Controlled DBC

Fix a, b, ℓ

1. Determine the interval of integers that can be represented with those chains

2. Enumerate the total number of DBCs with leading factor $2^a 3^b$ and length ℓ

Controlled DBC

Fix a, b, ℓ

1. Determine the interval of integers that can be represented with those chains

2. Enumerate the total number of DBCs with leading factor $2^a 3^b$ and length ℓ

3. Estimate the redundancy, i.e. how many chains represent the same integer on average

1. Interval

Any DBC with leading factor 2^a3^b belongs to the interval

$$\left[\frac{3^b+1}{2},\ 2^{a+1}3^b - \frac{3^b+1}{2}\right]$$

2. Enumeration

Definition.

Let $S_\ell(a, b)$ denote the number of unsigned DBCs of length ℓ with a leading factor equal to 2^a3^b

The quantity we are interested in is $2^{\ell-1}S_\ell(a, b)$

2. Enumeration

Definition.

Let $S_\ell(a, b)$ denote the number of unsigned DBCs of length ℓ with a leading factor equal to $2^a 3^b$

The quantity we are interested in is $2^{\ell-1} S_\ell(a, b)$

$$2^a 3^b + 2^{a_2} 3^{b_2} + \cdots + 2^{a_\ell} 3^{b_\ell}$$

2. Enumeration

Definition.

Let $S_\ell(a,b)$ denote the number of unsigned DBCs of length ℓ with a leading factor equal to 2^a3^b

The quantity we are interested in is $2^{\ell-1}S_\ell(a,b)$

$$2^a3^b \pm 2^{a_2}3^{b_2} \pm \cdots \pm 2^{a_\ell}3^{b_\ell}$$

2. Enumeration

We introduce $T_\ell(a,b)$ the number of unsigned DBCs of length ℓ with a leading factor dividing $2^a 3^b$

We oberve that

$$S_{\ell+1}(a,b) = T_\ell(a,b) - S_\ell(a,b)$$

$$T_{\ell+1}(a,b) = \sum_{i=0}^{a}\sum_{j=0}^{b}\left[(a-i+1)(b-j+1)-1\right]S_\ell(i,j)$$

2. Enumeration

We also have

$$S_1(a,b) = 1 \quad \text{and} \quad T_1(a,b) = (a+1)(b+1)$$

2. Enumeration

We also have

$$S_1(a,b) = 1 \quad \text{and} \quad T_1(a,b) = (a+1)(b+1)$$

Together with the last two equations, these relations allow to compute $S_\ell(a,b)$ recursively for any tuple (a,b,ℓ)

The actual computation can be carried out efficiently using some precomputations and Lagrange interpolation

3. Redundancy

The most difficult part is to estimate the redundancy of DBCs

We have run simulations and deduced heuristics

3. Redundancy

For $a \leqslant 30$, $b \leqslant 12$ and $\ell \leqslant 12$ we have computed the average number of representations of an integer with a DBC having leading factor equal to $2^a 3^b$ and length ℓ

3. Redundancy

For $a \leqslant 30$, $b \leqslant 12$ and $\ell \leqslant 12$ we have computed the average number of representations of an integer with a DBC having leading factor equal to $2^a 3^b$ and length ℓ

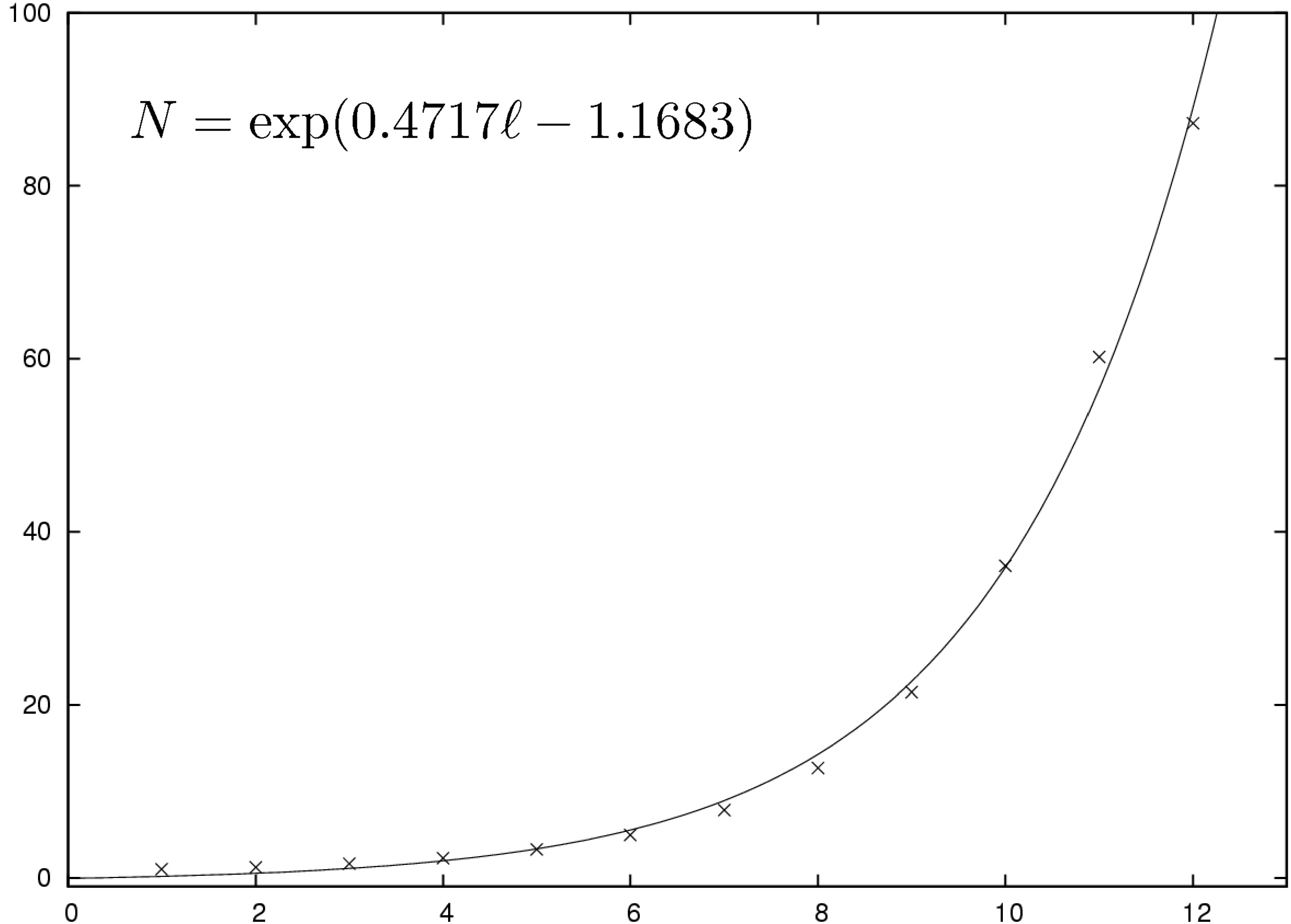
This was done with the algorithms explained in the first part

3. Redundancy

For $a \leqslant 30$, $b \leqslant 12$ and $\ell \leqslant 12$ we have computed the average number of representations of an integer with a DBC having leading factor equal to $2^a 3^b$ and length ℓ

This was done with the algorithms explained in the first part

The data fit an exponential regression of the form $N = \exp(0.4717\ell - 1.1683)$ with $R^2 = 0.9975$


$N = \exp(0.4717\ell - 1.1683)$
100
80
60
40
20
0
0
2
4
6
8
10
12

Near Optimal Length

Take a leading factor equal to $2^a 3^b \simeq 2^t$

Definition.

The Near Optimal Length is the value of ℓ minimizing

$$\left| 2^{\ell-1} S_\ell(a,b) - 2^t \lceil \exp(0.4717\ell - 1.1683) \rceil \right|$$

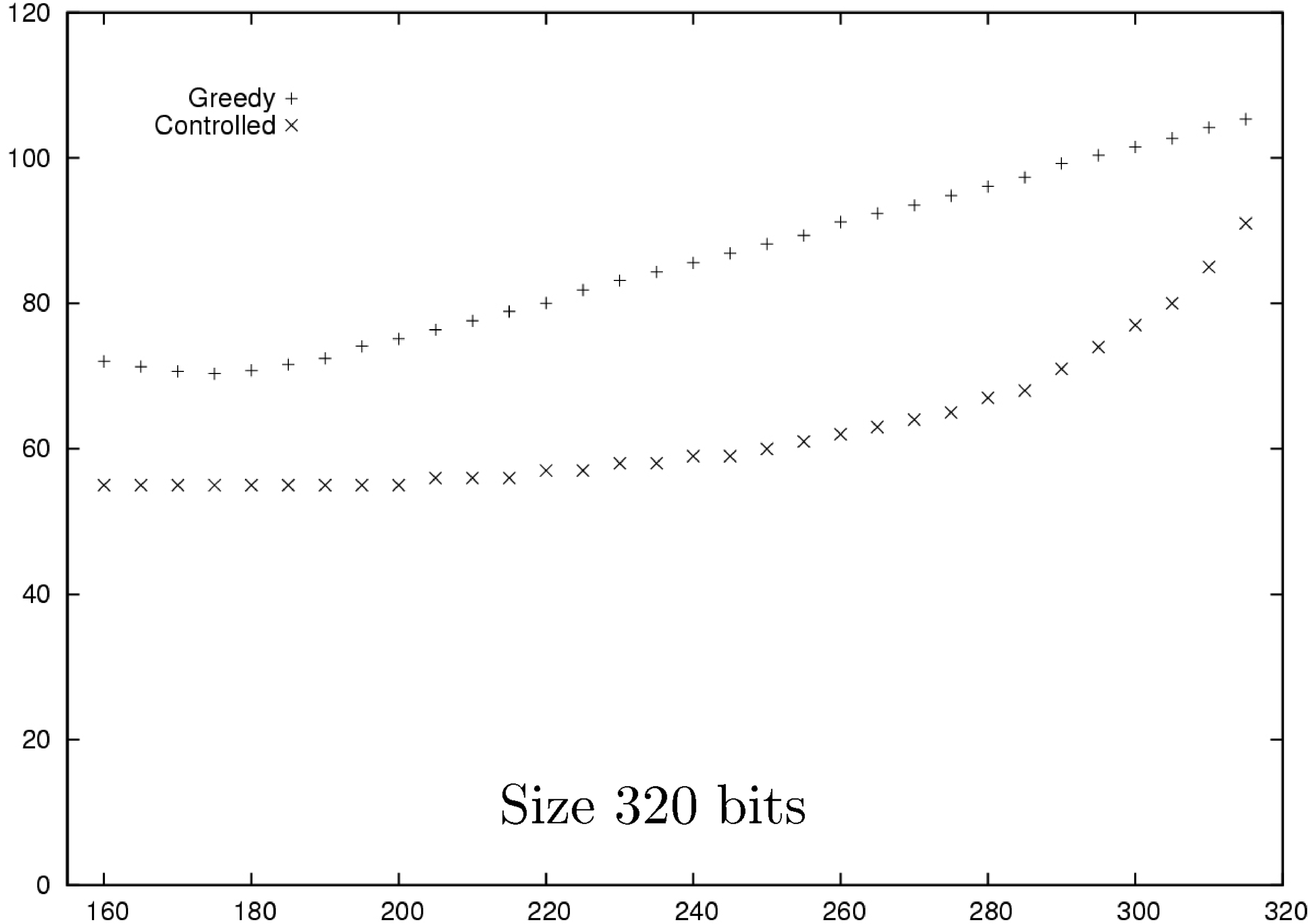
Comparison with Greedy

Consider scalar of size t bits, fix a between $t/2$ and t and consider the corresponding b $(2^a 3^b \simeq 2^t)$

Compute the Near Optimal Length

Compare with the average length of the DBCs returned by the greedy method

The Near Optimal Length is 20 to 30% shorter than Chains returned by the greedy method


Greedy +
Controlled ×
Size 320 bits
0
20
40
60
80
100
120
160
180
200
220
240
260
280
300
320

Near Optimal Controlled DBC

Given a particular coordinate system and a particular size of scalar

It is possible to derive the optimal parameters (a, b and ℓ) which minimizes the complexity of the scalar multiplication for that system

We have done that with Inverted Edwards coordinates

Near Optimal Controlled DBC

	Near Optimal			Greedy		
Size	LF	ℓ	Cost	LF	ℓ	Cost
192	$2^{151}3^{26}$	37	1570.20	$2^{116}3^{48}$	44.63	1688.74
256	$2^{198}3^{37}$	48	2092.60	$2^{153}3^{65}$	58.73	2249.62
320	$2^{260}3^{38}$	62	2612.40	$2^{180}3^{89}$	70.80	2816.04
384	$2^{297}3^{55}$	71	3128.40	$2^{217}3^{106}$	84.74	3375.51
448	$2^{369}3^{50}$	86	3645.80	$2^{254}3^{123}$	98.73	3935.42
512	$2^{406}3^{67}$	95	4161.80	$2^{286}3^{143}$	112.07	4495.22

Conclusion

We have enumerated the number of DBCs representing a given integer

This gives rise to a new method to find optimal DBCs

We have also enumerated the number of different DBCs with given parameters

This gives rise to a new scalar multiplication method where the scalar is selected in DBC format directly

Future Work

– Improve the optimal DBC algorithm (dynamic programming, pruning, etc)

– Analyze the redundancy of DBCs more precisely

– Given a, b and ℓ, return uniformly distributed DBCs with leading factor $2^a 3^b$ and length ℓ

Questions