Bounds and Trade-offs for Double-Base Number Systems*

Tiancheng Lou, Xiaoming Sun and Christophe Tartary

Institute for Interdisciplinary Information Sciences
Institute for Theoretical Computer Science
Tsinghua University
Beijing, 100084
People's Republic of China

loutiancheng860214@gmail.com
{xiaomings,ctartary}@mail.tsinghua.edu.cn

Abstract

During the past twenty years, elliptic curves have attracted more and more attention from the cryptography community. One of the core operations of elliptic curve cryptography protocols is the scalar multiplication. Any algorithm reducing the complexity of such multiplications will speed up cryptographic primitives based on these algebraic structures. In this paper, we study two recently introduced techniques for scalar multiplication: Double-Base Number System (DBNS) and Double-Base Chain (DBC). Our results are twofold. First, we demonstrate a theoretical bound $\Omega(\log n)$ on the length of any DBC used to decompose some $(\log n)$-bit integers. Second, we present a new algorithm to obtain a $\{2,3\}$-integer expansion of n. The bound previously computed will imply the optimality of this algorithm. Our scheme represents a trade-off method between the DBNS approach and the DBC technique. Experimentally, our algorithm constructs shorter chains on average than both the binary/ternary method and the greedy algorithm approach.

Keywords: Elliptic Curve Cryptography, Scalar Multiplication, Double-Base Number System, Double-Base Chain.

1 Introduction

The first elliptic curve cryptographic schemes were independently proposed by Koblitz [Kob87] and Miller [Mil86] in 1985. The interest in these algebraic structures comes from the fact that there is no subexponential time algorithm known for solving the discrete logarithm problem on elliptic curves unlike the case of the multiplicative group of a finite field [van05]. Thus, one can use smaller security parameters for elliptic curve protocols than for ordinary discrete logarithm schemes while having the same level of security. This allows the implementation of cryptographic primitives on devices with limited memory capacity such as smart cards [CFA+06].

The fundamental operation in elliptic curve cryptography is the scalar multiplication: given a point P on the curve and an integer n, one wants to compute the point $[n]P$. This operation has beenthe subject of intense research [CFA+06]. One of the central points is to represent n appropriately to perform fast curve operations. In 2005, Dimitrov *et al.* proposed to use the *Double-Base Number System* (DBNS) for cryptographic purposes [DIM05]. The integer n is expended into a sum of $\{2,3\}$-integers (i.e. integers written as $2^a\,3^b$). It was demonstrated that a greedy approach to expend n using DBNS had at most $O(\frac{\log n}{\log\log n})$ terms [DJM98]. Such an expansion is not unique and the greedy algorithm may return an expansion which requires extra storage to keep intermediate results. To overcome this issue, Dimitrov *et al.* introduced the concept of *Double-Base Chain* (DBC) [DIM05]. A DBC is a DBNS where the powers of 2 and 3 are both decreasing. This property simplifies the computation of the point $[n]P$.

In this paper, we first compute a lower-bound $\Omega(\log n)$ on the length of any DBC representation for some $(\log n)$-bit integers. We show that the length of the DBC returned by a modified greedy algorithm is optimal (tight). Secondly, we propose a new method to speed up the computation of $[n]P$. Our approach is based on the modification of DBCs that we call p-DBC.

The rest of the paper is organized as follows. In Section 2, we briefly recall some definitions about integers expansion and we present greedy algorithms for both DBNS and DBC. After proving the lower bound on the DBC length in Section 3, we present our trade-off method in Section 4 and some experimental results are discussed in Section 5. The last section concludes the paper.

*The original version of this paper appears in Information Processing Letters, vol. 111, no. 10, pp 488 - 493, April 2011, Elsevier.

2 Integer Expansion

In this section, we review some definitions and properties of DBNS and DBC. Both representations rely on the following operations on the elliptic curve:

- addition ($P + Q$) and subtraction ($P - Q$),
- doubling ($[2]P$) and tripling ($[3]P$).

Let n be a positive integer. In DBNS, the integer n is written as:

$$n = \sum_{i=1}^{\ell} c_i \, 2^{a_i} \, 3^{b_i} \text{ where } c_i \in \{-1, 1\} \tag{1}$$

One can construct a greedy algorithm to obtain a DBNS by finding at each step the best approximation of a particular integer as a $\{2, 3\}$-integer (Algorithm 1).

Algorithm 1 GreedyDBNS

Input: An integer n.

1. Initialize $t = n$, $p = 1$ and IntegerSet $= \emptyset$.
2. While $t \neq 0$ do the following:
   2.1. Find the best approximation of t as $z = 2^a 3^b$.
   2.2. Set IntegerSet $=$ IntegerSet $\cup \{p \cdot z\}$.
   2.3. If $t < z$, set $p = -p$.
   2.4. Set $t = |t - z|$.
3. Return IntegerSet.

Output: A DBNS expansion of n.

The length of the DBNS expansion of n is the size of IntegerSet. We have the following demonstrated by Dimitrov *et al.* based on a result from Tijdeman [Tij74].

Lemma 1 ([DJM98]) *For every positive integer n, the length ℓ of the* DBNS *expansion of n returned by Algorithm 1 is*

$$O\left(\frac{\log n}{\log \log n}\right)$$

Since the sequences of exponents may not simultaneously be decreasing, it may use

$$\sum_{i=1}^{\ell} (a_i + b_i) = O(\log n \cdot \ell) = O\left(\frac{\log^2 n}{\log \log n}\right)$$

doublings and triplings in the worst case.

In DBC, the integer n is written as in Equation (1) with the additional requirement:

$$a_\ell \geq a_{\ell-1} \geq \cdots \geq a_1 \text{ and } b_\ell \geq b_{\ell-1} \geq \cdots \geq b_1 \tag{2}$$

Using a DBC, one can compute $[n]P$ with a_ℓ doublings, b_ℓ triplings and $\ell - 1$ additions/subtractions. One can easily modify GreedyDBNS to get a greedy algorithm computing a DBC for n (Algorithm 2).

Lemma 2 *The length of the* DBC *expansion of n returned by Algorithm 2 is*

$$O(\log n)$$

It only uses

$$O(\log n + \ell) = O(\log n)$$

doubling and triplings.

Algorithm 2 GreedyDBC

Input: An integer n.

1. Initialize $t = n, p = 1, A = +\infty, B = +\infty$ and IntegerSet $= \emptyset$.
2. While $t \neq 0$ do the following:
   2.1. Find the best approximation of t as $z = 2^a 3^b$ where $0 \leq a \leq A$ and $0 \leq b \leq B$.
   2.2. Set IntegerSet $=$ IntegerSet $\cup \{p \cdot z\}$.
   2.3. If $t < z$, set $p = -p$.
   2.4. Set $t = |t - z|$, $A = a$ and $B = b$.
3. Return IntegerSet.

Output: A DBC expansion of n.

3 Lower Bound for DBCs

As said in Section 2, a DBC requires that the sequence of exponents be simultaneously decreasing. With this additional constraint, we can prove that, for any positive integer L, there exist L-bit integers for which any DBC representation requires $\Omega(L)$ terms. This implies that the bound stated in Lemma 2 for the algorithm GreedyDBC is tight. Before demonstrating this claim, we need some intermediate results.

Lemma 3 *For any n and m, we have: $\binom{n}{m} \leq \left(\frac{e\,n}{m}\right)^m$ where e is the value of the exponential function at* 1.

Lemma 4 *For any constant value c_1, there exists a constant value c_2, such that for any $n > 1$, we have:*

$$3^{c_2 \log_2 n} \binom{(c_1 + c_2) \log_2 n}{c_2 \log_2 n}^2 \leq n$$

Proof.
For any c_1, we choose $c_2 = \min\{c_1, \frac{1}{100}, \frac{1}{8 \log_2(2e \cdot c_1)}\}$. First, we have:

$$3^{c_2 \log_2 n} = n^{c_2 \log_2 3} \leq n^{1/2} = \sqrt{n}$$

Applying Lemma 3, we obtain:

$$\binom{(c_1 + c_2) \log_2 n}{c_2 \log_2 n}^2 \leq n^{2\, c_2\, (\log_2(2\, e\, c_1) - \log_2\, c_2)}$$

Since $c_2 \leq \frac{1}{8 \log_2(2\, e\, c_1)}$, we get:

$$2\, c_2\, \log_2(2\, e\, c_1) \leq \frac{1}{4}$$

For $c_2 \leq \frac{1}{100}$, we have:

$$-2 c_2 \log_2 c_2 \leq \frac{1}{4}$$

Pulling these results together, we obtain:

$$\binom{(c_1 + c_2) \log_2 n}{c_2 \log_2 n}^2 \leq n^{2\, c_2\, (\log_2(2\, e\, c_1) - \log_2\, c_2)} \leq \sqrt{n}$$

Finally:

$$3^{c_2\, \log_2\, n} \binom{(c_1 + c_2)\, \log_2\, n}{c_2\, \log_2\, n}^2 \leq \sqrt{n} \cdot \sqrt{n} \leq n$$

□

Lemma 5 *Let n be any positive integer. Suppose that its shortest* DBC *expansion is written as in Equation (1) with Condition (2). We have the following properties:*

1. $(a_i, b_i) \neq (a_{i+1}, b_{i+1})$, *for any* $1 \leq i \leq \ell - 1$

2. $c_\ell = 1$
3. $a_i \leq 2\log_2 n$, *for any* $1 \leq i \leq \ell$
4. $b_i \leq 2\log_2 n$, *for any* $1 \leq i \leq \ell$

Proof.
Property 1. Assume that there exists i such that $(a_i, b_i) = (a_{i+1}, b_{i+1})$. If $c_i = c_{i+1}$, we can use $(a_i + 1, b_i)$ instead of (a_i, b_i) and (a_{i+1}, b_{i+1}) to obtain a shorter expansion. Otherwise, $c_i \neq c_{i+1}$. In this case, we can delete both (a_i, b_i) and (a_{i+1}, b_{i+1}) to get a shorter expansion since $c_i = -c_{i+1}$.

Property 2. Applying Property 1, we have: $2^{a_{i+1}}3^{b_{i+1}} \geq 2 \cdot 2^{a_i}3^{b_i}$. We deduce: $2^{a_\ell}\,3^{b_\ell} \geq \sum_{i=1}^{\ell-1} 2^{a_i}\,3^{b_i}$. Since $n > 0$, the previous inequality implies $c_\ell = 1$.

Properties 3 and 4. If $\ell = 1$, then n is a $\{2,3\}$-integer and we have: $a_1 \leq \log_2 n$ and $b_1 \leq \log_3 n$. We now consider $\ell \geq 2$. We claim that if $c_{\ell-1} = -1$, then $2^{a_\ell}3^{b_\ell} \geq 3 \cdot 2^{a_{\ell-1}}\,3^{b_{\ell-1}}$. Indeed, in the opposite case, we would have: $a_{\ell-1} + 1 = a_\ell$ and $b_{\ell-1} = b_\ell$. Let k be the smallest number such that $c_k = c_{k+1} = \cdots = c_{\ell-1} = -1$, $a_k = a_{k+1} - 1 = a_{k+2} - 2 = \cdots = a_\ell - (\ell - k)$ and $b_k = b_{k+1} = \cdots = b_\ell$.

$$\begin{aligned}
\sum_{j=k}^{\ell} c_j\,2^{a_j}\,3^{b_j} &= -\left(\sum_{j=k}^{\ell-1} 2^{a_k+j-k}\,3^{b_k}\right) + 2^{a_k+\ell-k}\,3^{b_k} \\
&= -2^{a_k}\,3^{b_k}\sum_{j=k}^{\ell-1} 2^{j-k} + 2^{a_k+\ell-k}\,3^{b_k} \\
&= -2^{a_k}\,3^{b_k}(2^{\ell-k} - 1) + 2^{a_k+\ell-k}\,3^{b_k} \\
&= 2^{a_k}\,3^{b_k}
\end{aligned}$$

We can use $2^{a_k}\,3^{b_k}$ instead of the sum of $(\ell-k+1)$ $\{2,3\}$-integers to get a shorter expansion for n (note that $\ell-k+1 \geq 2$). This result would contradict the assumption made when enunciating Lemma (5) on the minimality of the DBC expansion of n.

We deduce that, when $c_{\ell-1} = -1$, we have:

$$n = \sum_{i=1}^{\ell} c_i\,2^{a_i}\,3^{b_i} \geq \frac{1}{3}\,2^{a_\ell}\,3^{b_\ell}$$

As a consequence, we get: $2^{a_\ell}\,3^{b_\ell} \leq 3\,n$. This implies Property 3 and Property 4 (when $c_{\ell-1} = -1$).

Assume that $c_{\ell-1} = 1$. We have:

$$n = \sum_{i=1}^{\ell} c_i\,2^{a_i}\,3^{b_i} \geq 2^{a_\ell}\,3^{b_\ell}$$

Since $2^{a_\ell}\,3^{b_\ell} \leq n$, Property 3 and Property 4 follow (when $c_{\ell-1} = 1$).

□

We can now demonstrate our lower bound on the length of DBCs.

Theorem 1 *For any positive integer L, there exist L-bit integers for which any DBC representation requires $\Omega(L)$ terms.*

Proof.
Let L be a positive integer. In order to apply the previous results, we set: $n = 2^L - 1$.

We choose $c_1 = 2$. According to Lemma (4), there exists a constant c_2 such that:

$$3^{c_2\,\log_2 n}\binom{(2+c_2)\,\log_2 n}{c_2\,\log_2 n}^2 \leq n$$

Due to Lemma (5), for any $1 \leq \ell \leq c_2\,\log_2 n$, the number of different DBC expansions having length ℓ does not exceed

$$2^{\ell-1}\binom{2\,\log_2 n + \ell - 1}{\ell}^2$$

because there are $2^{\ell-1}$ possibilities for the $(\ell-1)$-tuple $(c_1,\ldots,c_{\ell-1})$ and at most $\binom{2\log_2 n+\ell-1}{\ell}$ different increasing families $0 \leq a_1 \leq \cdots \leq a_\ell < 2\log_2 n$.

We have the following relation (parallel summation [GKP94]):

$$\sum_{\ell=0}^{c_2 \log_2 n} \binom{2\log_2 n+\ell-1}{\ell} = \binom{(2+c_2)\log_2 n}{c_2 \log_2 n}$$

Therefore, the number of DBC expansions having $a_i, b_i \leq 2\log_2 n$ and containing no more than $c_2 \log_2 n$ terms does not exceed

$$\begin{aligned}
\sum_{\ell=1}^{c_2 \log_2 n} 2^{\ell-1} \binom{2\log_2 n+\ell-1}{\ell}^2 &< \sum_{\ell=1}^{c_2 \log_2 n} 2^{\ell-1} \binom{2\log_2 n+\ell}{\ell}^2 \\
&< 2^{c_2 \log_2 n-1} \binom{(2+c_2)\log_2 n}{c_2 \log_2 n}^2 \\
&< \frac{1}{2}\left(\frac{2}{3}\right)^{c_2 \log_2 n} n \\
&< \frac{1}{2} \times \frac{2}{3} \times n \\
&< \frac{n}{3}
\end{aligned}$$

Thus, using at most $c_2 \log_2 n$ terms is not sufficient to represent a third of all integers in $\{1, 2, \cdots, n = 2^L - 1\}$. The set of L-bit integers is $\{2^{L-1}, \ldots, 2^L - 1\}$ and contains $2^{L-1} > \frac{n}{3}$ values. Therefore, at least one of these L-bit elements cannot have any DBC representations using at most $c_2 \log_2 n = \Omega(L)$ terms.

□

4 Trade-off Method

Let n be a positive integer. We call a *p-Double Base Chain* (p-DBC) expansion of n, any representation of n as in Equation (1) with the following condition:

$$\forall i \in \{1,\ldots,\ell-1\} \begin{cases} 2^{a_i} 3^{b_i} \leq 2^{a_{i+1}} 3^{b_{i+1}} \\ a_i \leq \min\limits_{k \geq i+1}\{a_k\} + p \\ b_i \leq \min\limits_{k \geq i+1}\{b_k\} + p \end{cases}$$

where ℓ represents the length of the expansion and $p \in \mathbb{N}$.

The algorithm GreedyDBC can easily be modified to compute such a p-DBC (Algorithm (3)).

Algorithm 3 p-GreedyDBC

Input: An integer n and the parameter p.

1. Initialize $t = n, op = 1, A = +\infty, B = +\infty$ and IntegerSet $= \emptyset$.
2. While $t \neq 0$ do the following:
   2.1. Find the best approximation of t as $z = 2^a 3^b$ where $0 \leq a \leq A + p$ and $0 \leq b \leq B + p$.
   2.2. Set IntegerSet $=$ IntegerSet $\cup \{op \cdot z\}$.
   2.3. If $t < z$, set $op = -op$.
   2.4. Set $t = |t - z|, A = \min\{A, a\}$ and $B = \min\{B, b\}$.
3. Return IntegerSet.

Output: A p-DBC expansion of n.

We now state our main result for this section.

Theorem 2 *The length ℓ of the p-DBC($p > 1$) expansion of n returned by Algorithm (3) is*

$$O\left(\frac{\log n}{\log p}\right)$$

It uses

$$O(\log n + p\,\ell) = O\left(\log n + \frac{p \log n}{\log p}\right)$$

doubling and triplings.

Before demonstrating this theorem, we need to prove the following lemma.

Lemma 6 *There exists a constant $C > 0$ such that, for any $n \leq 2^A 3^B$, there is always a number of the form $2^a 3^b$ between $n - n/p^C$ and n, where $a \leq A + p$ and $b \leq B + p$.*

Proof.
To prove this lemma, we only need to consider the case when $A + B > \lfloor p/3 \rfloor$ and $2^{A-1}3^B \leq n \leq 2^A 3^B$. Because when $A + B \leq \lfloor p/3 \rfloor$, using the result by Dimitrov *et al.*, we can prove that there is always a number of the form $2^a 3^b$ between $n - n/(\log n)^C$ and n.

When $p \leq 5$, we can choose $C = \log_5 2$, such that $p^C \leq 2$ and $2^{A-1}3^B \geq n/2 \geq n - n/p^C$, so $2^{A-1}3^B$ is between $n - n/p^C$ and n. It remains to prove the lemma when $p \geq 6$. We need the following result by Tijdeman.

Lemma 7 *[Tij74] There exists a constant $C > 0$ such that, for any N, there is a number of the form $2^a 3^b$ between $N - N/(\log N)^C$ and N.*

According to Lemma (7), there exists an constant $C > 0$, such that if we set $A' = \max\{0, A - \lfloor p/3 \rfloor\}$, $B' = \max\{0, B - \lfloor p/3 \rfloor\}$ and $y = n/\left(2^{A'}3^{B'}\right)$, there exists $x = 2^a 3^b$ satisfying

$$y - y/(\log y)^C \leq x \leq y.$$

Consider $a' = A' + a$ and $b' = B' + b$. We have:

$$2^{a'}3^{b'} \leq x \cdot 2^{A'}3^{B'} \leq y \cdot 2^{A'}3^{B'} \leq n$$

Since $n \geq 2^{A-1}3^B \geq 1/2 \cdot 2^A 3^B$, we have

$$y \geq n/\left(2^{A'}3^{B'}\right) \geq 1/2 \cdot 2^{A-A'}3^{B-B'}.$$

Since $A + B > \lfloor p/3 \rfloor$, we obtain:

$$\log_2 y \geq \log_2\left(1/2 \cdot 2^{\lfloor p/3 \rfloor}\right) \geq \lfloor p/3 \rfloor - 1$$

The previous inequality implies $\log_2 y = \Omega(p)$. Therefore, there exists a constant $0 < C' < 1$ such that, for any $p \geq 6$, we have: $\log_2 y \geq p^{C'}$. We get:

$$x \geq y - y/(\log y)^C \geq y - y/p^{C'C}$$

We deduce:

$$2^{a'}\,3^{b'} \geq (y - y/p^{C'C})2^{A-\lfloor p/3 \rfloor}3^{B-\lfloor p/3 \rfloor} \geq n - n/p^{C'C}$$

Since $x \leq y \leq 2^{\lfloor p/3 \rfloor}3^{\lfloor p/3 \rfloor} \leq 2^p$, we have $a \leq p$ and $b \leq p$. Therefore, the relations $a' \leq A + p$, $b' \leq B + p$ and $n - n/p^{C'C} \leq 2^{a'}3^{b'} \leq n$ are satisfied which terminates our proof.

□

We can now demonstrate Theorem (2).

Proof.
According to Lemma (6), the length of the p-DBC expansion of n returned by Algorithm (3) is at most

$$O\left(\log_{p^C} n\right) = O\left(\frac{\log n}{\log p}\right).$$

Suppose the p-DBC expansion of n returned by the greedy algorithm is $n = \sum_{i=1}^{\ell} c_i\, 2^{a_i}\, 3^{b_i}$. Denote the point $[2^{a_i}3^{b_i}]P$ by (a_i, b_i). We can use the following method to calculate all (a_i, b_i) pairs.

- Starting from $A = 0$, $B = 0$ and we have $(A, B) = [2^A\, 3^B]P = P$.
- Iterate i from 1 to ℓ.
  - Set $A' = \min_{k \geq i}\{a_k\}$ and $B' = \min_{k \geq i}\{b_k\}$.
  - Compute $(A', B') = [2^{A'}\, 3^{B'}]P$ and $(a_i, b_i) = [2^{a_i} 3^{b_i}]P$.
  - Set $A = A'$, $B = B'$ and $(A, B) = (A', B')$.

Remark: At each iteration, we have: $(A', B') = [2^A\, 3^B]([2^{A'-A}\, 3^{B'-B}]P) = 2^{A'-A}\, 3^{B'-B}\, (A, B)$ and $(a_i, b_i) = [2^{A'}\, 3^{B'}]([2^{a_i-A'}\, 3^{b_i-B'}]P) = 2^{a_i-A'}\, 3^{b_i-B'}\, (A', B')$.

If we set $a_0 = b_0 = 0$, then the number of doublings and triplings required by this method does not respectively exceed

$$\sum_{i=0}^{\ell-1} \left(\min_{k \geq i+1}\{a_k\} - \min_{k \geq i}\{a_k\} + \min_{k \geq i+1}\{b_k\} - \min_{k \geq i}\{b_k\} + 2\,p \right)$$

and

$$\min_{k \geq \ell}\{a_k\} - \min_{k \geq 0}\{a_k\} + \min_{k \geq \ell}\{b_k\} - \min_{k \geq 0}\{b_k\} = a_\ell + b_\ell + 2\,p\,\ell$$

We can note that $a_\ell \leq 2\, \log_2 n$ and $b_\ell \leq 2\, \log_2 n$. Therefore, the number of doublings and triplings are

$$O(\log n + p\,\ell) = O\left(\log n + \frac{p\, \log n}{\log p}\right)$$

□

The p-DBC expansion technique is indeed a trade-off method between DBNS and DBC.

- If $p = O(1)$, the length of expansion is

  $$O\left(\frac{\log n}{\log p}\right) = O(\log n)$$

  The number of doublings and triplings are

  $$O\left(\log n + \frac{p\, \log n}{\log p}\right) = O(\log n)$$

- If $p = O(\log n)$, then the length of expansion is

  $$O\left(\frac{\log n}{\log p}\right) = O\left(\frac{\log n}{\log \log n}\right)$$

  The number of doublings and triplings are

  $$O\left(\log n + \frac{p\, \log n}{\log p}\right) = O\left(\frac{\log^2 n}{\log \log n}\right)$$

We can apply the new method to compute such a p-DBCs. The method to compute DBCs can be seen as a generalization of the binary/ternary approach [DKS09]. As in Doche *et al.*'s work, we can use a tree-based approach [DIM05] for our method. This is depicted in Algorithm (4).

5 Experiments

In this part, we analyze the result of computations performed on random integers of various sizes and represented as Table 1. The first observation is that even when p is 0, the length of the DBC expansion returned by p-DBCTree is in general less than what is obtained from the greedy technique p-GreedyDBC. Table 1 also compares the lengths of DBCs obtained from various method: binary/ternary, greedy algorithm, and our new approach. The average length of chains returned by the tree-based algorithm is approximately 20% shorter than the length of the chains returned by the greedy method.

Algorithm 4 p-DBCTree

Input: An integer n and the parameter p.

1. Initialize $t = f(n)$, where $f(n) = n/\left(2^{v_2(n)}3^{v_3(n)}\right)$, and $v_k(n) = \max\{e|n \text{ is a multiple of } k^e.\}$.
2. While $t \neq 1$ do the following:
   2.1. Set $t' = t$.
   2.2. For a from 0 to p, do:
      For b from 0 to p, do:
         Set $t' = \min(t', f(t - 2^a\,3^b), f(t + 2^a\,3^b))$.
   2.3. Set $t = t'$.

Output: A p-DBC expansion of n.

Table 1: Parameters of DBCs obtained by various methods.

Bit Size of n	256			320			384			448			512		
Method	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ	ℓ	a_ℓ	b_ℓ
Binary/Ternary	58.3	116.2	87.4	72.9	145.5	108.6	87.2	175.2	131.2	101.8	204.4	152.2	116.5	233.5	175.2
Greedy	58.3	150.7	87.1	72.9	189.2	81.5	87.0	228.0	97.3	101.3	266.3	113.7	115.7	305.0	129.5
Our Method	53.37	140.0	70.3	67.70	175.3	89.2	81.34	210.9	107.3	96.04	250.1	123.3	109.69	291.1	137.9

We also studied the impact of p on the average length of the chains (Figure 1). For each bit length of n (namely $256, 320, 384, 448, 512$), we ran the tree-based algorithm on 10,000 random integers with different choices of p, namely $p = 0, 1, 2, 3, 4, 5$ and 6.

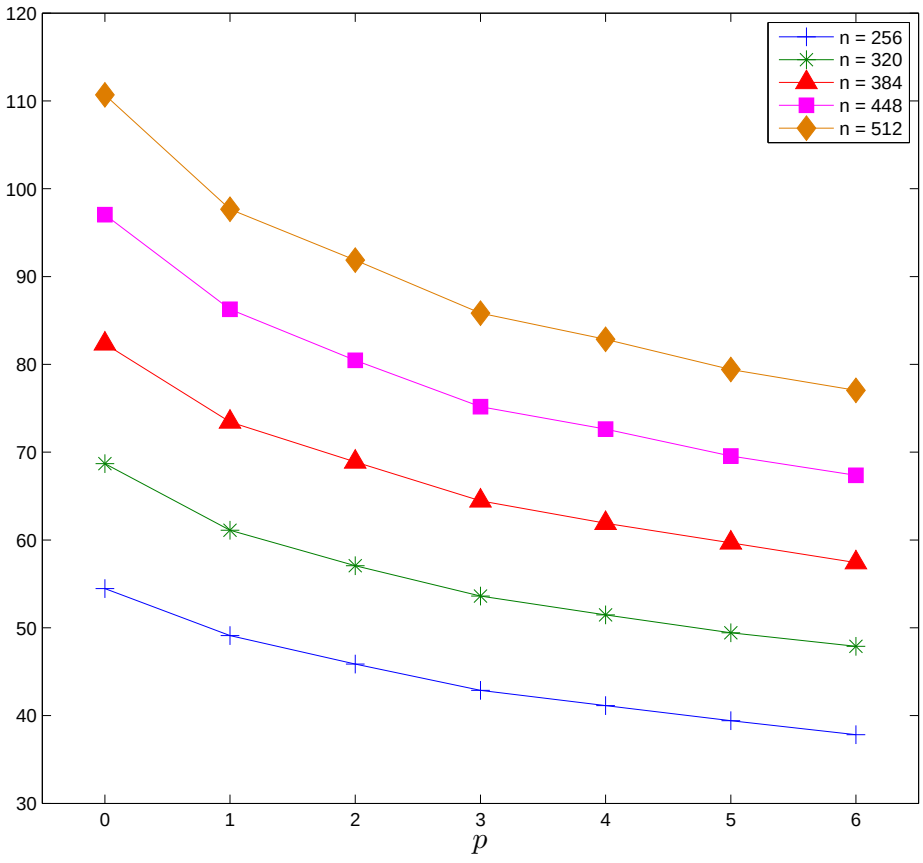

Figure 1: Impact of p on the average length of p-DBCs

These implementations showed the decrease in the length ℓ of the integer expansion as expected.

6 Conclusion

In this paper, we first studied the properties of DBNS and DBC. We constructed a lower bound on the length of DBCs which implied the tightness of the greedy algorithm technique to construct them. Second, we introduced a new integer expansion technique called p-DBC representation. This new kind of expansion is a trade-off method between DBNS and DBC. We proved some bounds on the length of p-DBCs. We conducted some implementations which confirmed that the those chains were shorter on average than DBCs constructed either by the binary/ternary method or the greedy algorithm approach.

Acknowledgments

We would like to thank Christophe Doche for discussion and comments on this work. This work was supported by the National Natural Science Foundation of China grant 60553001 and the National Basic Research Program of China grants 2007CB807900 and 2007CB807901. Xiaoming Sun's research was also funded by the National Natural Science Foundation of China grant 60603005, 60621062 and Tsinghua University Initiative Scientific Research Program 2009THZ02120. Christophe Tartary's work is also financed by the National Natural Science Foundation of China under the International Young Scientists program (grant 61050110147).

References

[CFA+06] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete Mathematics and Its Applications. Chapman & Hall/CRC, 2006.

[DIM05] Vassil Dimitrov, Laurent Imbert, and Pradeep Kumar Mishra. Efficient and secure elliptic curve point multiplication using double-base chains. In Bimal K. Roy, editor, *Advances in Cryptology - Asiacrypt'05*, volume 3788 of *Lecture Notes in Computer Science*, pages 59–78, Chenai, India, December 2005. Springer - Verlag.

[DJM98] Vassil S. Dimitrov, Graham A. Jullien, and William C. Miller. An algorithm for modular exponentiation. *Information Processing Letters*, 66(3):155 – 159, May 1998.

[DKS09] Christophe Doche, David R. Kohel, and Francesco Sica. Double-base number system for multi-scalar multiplications. In Antoine Joux, editor, *Advances in Cryptology - Eurocrypt'09*, volume 5479 of *Lecture Notes in Computer Science*, pages 502 – 517, Cologne, Germany, May 2009. Springer.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, second edition, 1994.

[Kob87] Neil Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203 – 209, January 1987.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology - Crypto'85*, volume 218 of *Lecture Notes in Computer Science*, pages 417 – 426, Santa Barbara, USA, August 1986. Springer - Verlag.

[Tij74] Robert Tijdeman. On the maximal distance between integers composed of small primes. *Compositio Mathematica*, 28(2):159 – 162, 1974.

[van05] Henk C. A. van Tilborg, editor. *Encyclopedia of Cryptography and Security*. Springer, 2005.