УДК 004.22

РАЗРАБОТКА НЕЙРОСЕТЕВОГО МЕТОДА УМНОЖЕНИЯ ТОЧКИ ЭЛЛИПТИЧЕСКОЙ КРИВОЙ НА СКАЛЯР

Н. И. ЧЕРВЯКОВ
Е. С. КИЯШКО

ФГАОУ ВПО «Северо-Кавказский федеральный университет»

e-mail: e.sergeevna.fmf@mail.ru

В статье предложен новый модифицированный метод вычисления скалярного умножения с использованием нейронной сети конечного кольца. В первую очередь исследования в данной области направлены на снижение алгоритмической сложности алгоритмов и, вместе с тем, на увеличение их быстродействия. С целью повышения эффективности предлагается использование нейронной сети конечного кольца и модифицированного NAF – non-adjacent form метода[2].

Ключевые слова: эллиптическая кривая, система остаточных классов, нейронные сети, умножение точки эллиптической кривой на скаляр, эллиптическая криптография, проективные координаты.

Введение. Постановка задачи

На современном этапе развития общества требуются особые подходы для сохранения информации в секрете, так как порой от информации зависят не только деньги, но и жизни людей. Развитие распределенных систем обработки данных выдвинуло на первый план задачу информационной безопасности системы в целом, понимая под этим состояние защищенности всех процессов обработки, хранения и передачи информации в системе [16]. Одним из перспективных направлений построения криптографических примитивов является эллиптическая кривая, так как она обеспечивает максимально возможный уровень безопасности при сохранении той же длины ключа, например, при длине ключа 192 бит криптосистема на эллиптической кривой обеспечивает тот же уровень безопасности RSA с длиной ключа 1024 бита. В данной работе будет рассматриваться эллиптическая кривая в форме Вейерштрасса: $E(F_p): y^2 = x^3 + ax + b$, где $a, b \in F_p$, $p > 3$ – простое число.

В основе криптосистем, построенных на точках эллиптической кривой, лежит алгоритм умножения точки на скаляр. На практике используются длинные криптографические ключи, что приводит к решению задач повышенной размерности, эффективное решение которых можно осуществить на основе использования нейросетевых систем, синтезируемых на основе нейроускорителей [19], представляющих собой совокупность нейронных сетей (НС) конечного кольца [18] или обычных арифметических элементов, которые выполняют арифметические операции. Ускорение выполнение базовых операций с точками эллиптической кривой получено в работах [14, 13] с использованием системы остаточных классов. В работе [15, 17] показано, что нейронная сеть представляет собой высокопараллельную динамическую структуру с топологией направленного графа, которая может получать выходную информацию посредством реакции её состояния на входные воздействия. Структура алгоритма обработки данных, представленных в системе остаточных классов, также как и структура нейронной сети обладают естественным параллелизмом, что позволяет использовать нейронную сеть в качестве аппарата описания алгоритма.

В данной работе проведен анализ известных методов и алгоритмов вычисления скалярного умножения точки на число, а также предложен новый модифицированный нейросетевой метод вычисления скалярного умножения с использованием нейронной сети конечного кольца.

Анализ новых эффективных алгоритмов сложения, удвоения и вычисления кратных точек на эллиптических кривых над конечными простыми полями характеристики больше 3 ,показал, что такого рода алгоритмы необходимы при реализации криптосистем, использующих группы точек эллиптических кривых над конечными полями, в том числе таких, которые основаны на применении спариваний

Вейля и Тейта. Традиционные методы умножения на скаляр, такие как использование проективных координат и «оконных» методов, являются не достаточно эффективными для спариваний Вейля и Тейта[11]. Особое внимание уделено вычислению кратных точек, так как данная операция является одной из самых основных операций для точек эллиптической кривой, которая значительно повышает безопасность при реализации программы, особенно в встраиваемых устройствах.

Основная часть

Операция умножения на скаляр в аддитивных группах происходит по аналогии с алгоритмом возведения в степень в мультипликативных группах. Однако, некоторые различия при умножении на скаляр все же существуют. Например, точки инверсии в группе точек ЭК позволяют оптимизировать количество операций в алгоритме. Кроме того, многочисленное сложение точек можно рассматривать как умножение на скаляр, что приводит к повышению скорости выполнения операций.

В работе [9] проанализированы простые бинарные методы, которые известны как алгоритмы удвоения и сложения точек ЭК в разных системах координат и приведена оценка их сложности вычисления.

При этом были использованы следующие обозначения: деление в поле, содержащем координаты точки; инверсия I, умножение M в $GF(p)$, возведения в квадрат S.

Сложность вычисления операций сложения и умножения представлена, в табл. 1.

Таблица 1

Сложность вычисления операций сложения и умножения

Система координат	Сложность вычисления сложения точек ЭК	Сложность вычисления удвоения точек ЭК
Аффинная (A)	$I+2M+S$	$I+2M+2S$
Проективная (P)	$12M+2S$	$7M+5S$
Якобиевы (J)	$12M+4S$	$4M+6S$
Якоби-Чудновского (J^c)	$11M+3S$	$5M+6S$
Модифицированные координаты Якоби (J^m)	$13M+6S$	$4M+4S$
Упрощенные координаты Якоби-Чудковского (J^s)	$12M+3S$	$4M+6S$

Таким образом, из приведенных данных в таблице 1 можно сделать следующие выводы:

1. Из всех рассматриваемых координат в Якоби-Чудновского операция сложения двух разных точек эллиптической кривой выполняется быстрее всего.
2. Наиболее быстрым из всех рассматриваемых координат является удвоение в модифицированных координатах Якоби.
3. Распараллелив вычисление кратной точки на несколько вычислительных потоков можно наиболее эффективно использовать модифицированные координаты Якоби.

Формулы (1) и (2) показывают два алгоритма умножения на скаляр, представленного в бинарном виде. Умножение происходит аналогично классического алгоритма square-and-multiply[12]:

$$\text{Алгоритм 1. } kP = k_0P + k_1 2P + \ldots + k_{l-1}2^{l-1}P; \quad (1)$$

$$\text{Алгоритм 2. } kP = k_0P + 2(k_1P + 2(\ldots + 2(k_{l-1}P)\ldots)). \quad (2)$$

Алгоритм 1 использует скаляр от наибольшего значения до наименьшего, так называемый подход left-to-right, а Алгоритм 2 наоборот – right-to-left.

Алгоритм 1. left-to-right сложение и удвоение точек при умножении на скаляр.

Вход: $P \in E(F_q), k = (k_{l-1}k_{l-2}..k_0)_2$

Выход: $G = kP$
1: $Q \leftarrow O$
2: **for** $i = l-1$ **to** 0 **do**
3: $Q \leftarrow 2Q$
4: **if** $k_i = 1$ **then**
5: $Q \leftarrow Q + P$
6: **return** Q

Алгоритм 2. right-to-left сложение и удвоение точек при умножении на скаляр.
Вход: $P \in E(F_q), k = (k_{l-1}k_{l-2}..k_0)_2$
Выход: $G = kP$
1: $Q \leftarrow O$
2: $R \leftarrow P$
3: **for** $i = 0$ **to** $l-1$ **do**
4: **if** $k_i = 1$ **then**
5: $Q \leftarrow Q + R$
6: $R \leftarrow 2R$
7: **return** Q

Трудоемкость двух алгоритмов одинакова в отношении операций над точками, тем не менее существуют различия при умножении. С одной стороны вариант left-to-right позволяет использовать смешанные аффинно-проективные координаты для сложения точек. С другой стороны используя вариант алгоритма right -to- left мы можем выразить через точку Q в Якобиан координатах и точку R в координатах модифицированного Якобиана[5]. Для алгоритма 2, введены следующие обозначения J / J^m.

Смешанный Якобиан и модифицированный Якобиан были первоначально введены, чтобы ускорить процесс удвоения в алгоритме left-to-right умножения на скаляр. В работе [1] Кохен и соавторы различают три вида операций left-to-right умножения на скаляр: промежуточные удвоения, окончательное удвоение, сложение. Окончательное удвоение, это удвоение предшествующей операции сложения, а все остальные удвоения называются промежуточными. Обратим внимание, на тот факт, что использование модифицированного Якобиана позволяет экономить $1M + 1A$ операций, по сравнению с обычными проективными координатами. Следовательно, скорость удвоения будет $3M + 4S + 11A$. Обозначим данный подход J^m / J.

Таблица 2

Средняя скорость одного скалярного умножения Алгоритма 1 в зависимости от точки представления

Система координат	Скорость при случайном a	Скорость при $a = -3$
H	$11.5M + 6S + 13.5A$	$11.5M + 4S + 14.5A$
J	$8M + 7.5S + 14.5A$	$8M + 5.5S + 15.5A$
J^m / J	$8M + 6.5S + 15A$	

Таблица 3

Средняя скорость одного скалярного умножения Алгоритма 2 в зависимости от точки представления

Система координат	Скорость при случайном a	Скорость при $a = -3$
H	$13M + 6S + 13.5A$	$13M + 4S + 14.5A$
J / J^m	$10M + 6S + 15.5A$	

Оценка средней скорости вычисления Алгоритма 1 и, соответственно, алгоритма 2, в зависимости от скорости модульного умножения (M), модульного возведения в квадрат (S), и модульного сложения или вычитания (A), для различных проективных систем координат, представлена в Таблице 2 и Таблица 3, соответственно.

Из проведенного анализа алгоритмов вычисления удвоения и сложения точек эллиптической кривой, результаты которых представлены в таблицах 2 и 3, видно, что использование модифицированного Якобиана позволяет уменьшить количество операций умножения, сложения и возведения в квадрат над большими числами и минимизировать сложность алгоритмов вычисления умножения точки на скаляр.

Использование составных операций

Можно отметить, что в алгоритме left-to-right, точка Q переходит в $2Q+P$, при этом обрабатывается 1 бит. Таким образом, основной цикл можно записать в виде

for $i=l-1$ **to** 0 **do**
if $k_i=0$ **then**
$Q\leftarrow 2Q$
else
$Q\leftarrow 2Q+P$.

Использование модифицированного Якобиана со средней скоростью алгоритма 1 составляет: $8.5M+5.5S+12A$ в общем случае или $8.5M+4.5S+12.5A$ при $a=-3$, что позволяет увеличить скорость выполнения арифметических операций с точками эллиптической кривой, соответственно, 9-11% и на 4-7%.

Для вычисления точки $-P=(x,p-y)$ от точки $P=(x,y)$ требуется только одно вычитание двух чисел, что является менее трудоемкой операцией, по сравнению со сложением и удвоением точек эллиптической кривой, которое позволяет уменьшить вычислительную сложность алгоритма left-to-rigth за счет представления скаляра не в двоичной системе счисления, а в NAF.

Бинарный алгоритм NAF определяется следующим образом: NAF представляется в виде целого числа $k\in N^*$ с $(k_{l-1}k_{l-2}\ldots k_0)_{NAF}$, где $k_i\in\{-1,0,1\},0\le i<l-1$ и $k_{l-1}=1$, так что для всех пар последовательных цифр k_i и k_{i+1} мы имеем $k_ik_{i+1}=0$.

Алгоритм NAF обладает следующими свойствами:

1. Данный $k\in N^*$, $(k)_{NAF}$ является уникальным.

2. Длина представления NAF k имеет вид $[\log_2(k)]+1$ или $[\log_2(k)]+2$, то есть она имеет туже длину или на одну цифру больше, чем бинарное представление k.

3. Вес Хемминга – число ненулевых цифр от $(k)_{NAF}$ всегда минимальный из базиса 2 при использовании подписи для заданного k.

4. Средний вес Хэмминга в $l-$значном представлении для NAF будет примерно $l/3$.

Вычисление NAF положительного целого числа представлено в Алгоритме 3, который работает $O(l)$ и включает в себя только легкие операции.

Алгоритм 3. Вычисление NAF.
Вход: $k\in N^*$
Выход: $(k)_{NAF}$
1: $i\leftarrow 0$
2: while $k\ge 1$ **do**
3: if $k\bmod 2=1$ **then**
4: $k_i\leftarrow 2-(k\bmod 4)$
5: $k\leftarrow k-k_i$
6: **else**
7: $k_i\leftarrow 0$

8: $k \leftarrow k/2$
9: $i \leftarrow i+1$
10: **return** $(k_{l-1}k_{l-2}\ldots k_0)$

Алгоритм 4 показывает вычисления left-to-right скалярного умножения при разложения скаляра в NAF. Алгоритм 5, впервые предложенный Joye [5], является представлением варианта right-to-left в том числе, on-the-fly вычисление представления скаляра в NAF. Можно отметить, что left-to-right on-the-fly вычисление представления в NAF возможно также вычислить используя таблицу [8].

Алгоритм 4. left-to-right бинарное представление скалярного умножения в NAF.
Вход: $P \in E(F_q), k = (k_{l-1}k_{l-2}\ldots k_0)_{NAF}$
Выход: kP
Uses: P and Q
1: $Q \leftarrow O$
2: **for** $i = l-1$ **to** 0 **do**
3: $Q \leftarrow 2Q$
4: **if** $k_i = 1$ **then**
5: $Q \leftarrow Q+P$
6: **if** $k_i = -1$ **then**
7: $Q \leftarrow Q+(-P)$
8: **return** Q

Алгоритм 5. right -to- left бинарное представление скалярного умножения в NAF.
Вход: $P \in E(F_q), k \in N^*$
Выход: kP
Uses: Q **and** R
1: $Q \leftarrow O$
2: $R \leftarrow P$
3: **while** $k \geq 1$ **do**
4: **if** $k \bmod 2 = 1$ **then**
5: $u \leftarrow 2-(k \bmod 4)$
6: $k \leftarrow k-u$
7: **if** $u = 1$ **then**
8: $Q \leftarrow Q+R$
9: **else**
10: $Q \leftarrow Q+(-R)$
11: $k \leftarrow k/2$
12: $R \leftarrow 2R$
13: **return** Q

Таблица 4

Средняя скорость вычисления умножения на скаляр в алгоритме 4 в зависимости от выбора точки

Система координат	Скорость при случайном a	Скорость при $a = -3$
H	$10M+5.7S+12.3A$	$10M+3.7S+13.3A$
J	$6.7M+7S+13.3A$	$6.7M+5S+14.3A$
J^m / J	$6.7M+5.7S+14A$	

Таблица 5

Средняя скорость вычисления умножения на скаляр в алгоритме 5 в зависимости от выбора точки

Система координат	Скорость при случайном a	Скорость при $a=-3$
H	$11M+5.7S+12.3A$	$11M+3.7S+13.3A$
J/J^m	$8M+5.3S+14.3A$	

Как было описано выше варианты left-to-right и right -to- left имеют различную скорость вычисления. Средняя скорость вычисления алгоритмов 4 и 5 приведены в таблице 4 и 5, соответственно.

Применение алгоритмов left-to-right и right -to- left, позволяет уменьшить вычислительную сложность вычисления операций сложения и удвоения в системе различных проективных координат.

Алгоритм 4 может использовать операцию $2Q+P$ для ускорения вычислений, в том числе для бинарного алгоритма. В этом случае основной цикл записывается в виде:

for $i=l-1$ **to** 0 **do**
if $k_i=0$ **then**
$Q\leftarrow 2Q$
else if $k_i=1$ **then**
$Q\leftarrow 2Q+P$
else
$Q\leftarrow 2Q+(-P)$

Используя при сложении смешанный Якобиан средняя скорость Алгоритма 4 составит в общем случае $7M+5.7S+11.7A$ или $7M+4.3S+12.3A$ при $a=-3$. Среднее увеличение скорости составляет около 6-8% в общем случае и 3-5% при $a=-3$.

Классические m-ичные алгоритмы возведение в степень

Алгоритмами m-ичного умножения на скаляр, являются алгоритмы сложения и удвоения точек, которые могут быть представлены с основанием $m>2$ [12]. Обычно, для повышения эффективности во встраиваемых устройствах m является степенью. При $m=2^t$ поиск t скалярного бита требует времени и предварительных вычислений: $2P,3P,\ldots,(m-1)P$.

«Оконные» алгоритмы NAF

Путем предварительного вычисления нечетных чисел, кратных входной точке, можно повысить эффективность скалярного умножения. Например, при обработке 11 бит, в алгоритме left-to-right, удвоение и сложение точек выполняется следующим образом $2(2Q+P)+P$. Заметим, что можно вынести сложение, то есть $3P$, тогда вычисление будет иметь вид $2(2Q)+3P$. Отсюда следует при использовании больших размеров «окон» NAF представления может быть получен существенный выигрыш[3,6].

Более того, если базовая точка является постоянной или заранее известной, то предварительные расчеты можно проводить в автономном режиме. Так, используя алгоритм right-to-left, мы сохраняем несколько промежуточных результатов во время вычисления кратной точки и объединяем их в конце [10,7], данную операцию будем называть поствычислениями.

Рассмотрим два «оконных» алгоритма описанных Ханкерсоном, Менезесом и Ванстоуном [4]: «скользящее окно» NAF и ширина «окна-w» NAF. Каждый из них может быть реализован через алгоритмы left-to-right и right -to- left .

Алгоритм 6 реализует алгоритм «скользящее окно» NAF используя алгоритм left-to-right и принимает двоичное представление скаляра в качестве входных данных. Алгоритм 7 реализует алгоритм ширина «окна-w» NAF используя алгоритм right -to- left и вычисление on-the-fly ширины «окна-w» представлением k .

Алгоритм 6. Алгоритм нахождения кратной точки left-to-right «скользящего окна» NAF

Вход: $P \in E(F_q), k = (k_{l-1}k_{l-2}..k_0)_{NAF}, w \geq 2$

Выход: kP

Uses: $Q, P_1, P_3, \ldots,$ **and** P_m with $m = 2(2^w - (-1)^w)/3 - 1$

1: $Q \leftarrow O$

2: $i \leftarrow l - 1$

Precomputations

3: **for** $i = 1$ **to** m **by** 2 **do**

4: $P_i \leftarrow iP$

Основной цикл

5: **while** $i \geq 0$ **do**

6: **if** $k_i = 0$ **then**

7: $Q \leftarrow 2Q$

8: $i \leftarrow i - 1$

9: **else**

10: $s \leftarrow \max(i - w + 1, 0)$

11: **while** $k_s = 0$ **do**

12: $s \leftarrow s + 1$

13: $u \leftarrow (k_i..k_s)_{NAF}$

14: **for** $j = 1$ to $i - s + 1$ **do**

15: $Q \leftarrow 2Q$

16: **if** $u > 0$ **then**

17: $Q \leftarrow Q + P_u$

18: **if** $u < 0$ **then**

19: $Q \leftarrow Q + (-P_{-u})$

20: $i \leftarrow s - 1$

21: **return** Q

Алгоритм 7. Алгоритм нахождения кратной точки right-to-left ширина «окна-w» NAF

Вход: $P \in E(F_q), k = (k_{l-1}k_{l-2}..k_0), w \geq 2$

Выход: kP

Uses: $R, Q_1, Q_3, \ldots,$ **and** Q_m **with** $m = 2^{w-1} - 1$

1: $R \leftarrow P$

2: $Q_1, Q_3, \ldots, Q_m \leftarrow O$

Основной цикл

3: **while** $k \geq 1$ **do**

4: **if** $k \bmod 2 = 1$ **then**

5: $t \leftarrow k \bmod 2^w$

6: **if** $t > 0$ **then**

7: $Q_t \leftarrow Q_t + R$

8: **if** $t < 0$ **then**

9: $Q_{-t} \leftarrow Q_{-t} - R$

10: $k \leftarrow k - t$

11: $R \leftarrow 2R$

12: $k \leftarrow k/2$

Postcomputations

13: **for** $i = 3$ **to** m **by** 2 **do**

14: $Q_1 \leftarrow Q_1 + iQ_i$

15: **return** Q_1

Метод «окон» позволяет понизить вычислительную сложность алгоритма, за счет использования больших массивов данных, которые вычисляются заранее. Однако, стоит заметить, что использование метода окон в реальных приложениях затруднено за счет требуемых больших пред и пост вычислений.

Так как в криптографии используются числа большой размерности, то для эффективной реализации арифметических операций с длинными числами используем системы остаточных классов. Основой нейросетевых систем будут являться нейроускорители, представляющие собой совокупность нейронных сетей конечного кольца и обычных арифметических элементов, которые выполняют арифметические операции[17].

В архитектуре нейронных сетей для вычисления скалярного умножения точки на скаляр сборный слой разбит на две группы по n' и n'' нейронов. Группа с n' нейронами используется для сбора входов P_i, а группа с n'' – для входов r. Синаптические веса первой группы n' равны $W_i = kP \bmod p_i$. Синаптические веса для второй группы n'' равны $W_i = 2^i$, где $i = 0,\ldots,n'-1$. Топология нейронной сети реализующей операцию вычисления умножения точки на скаляр представленна на рисунке.

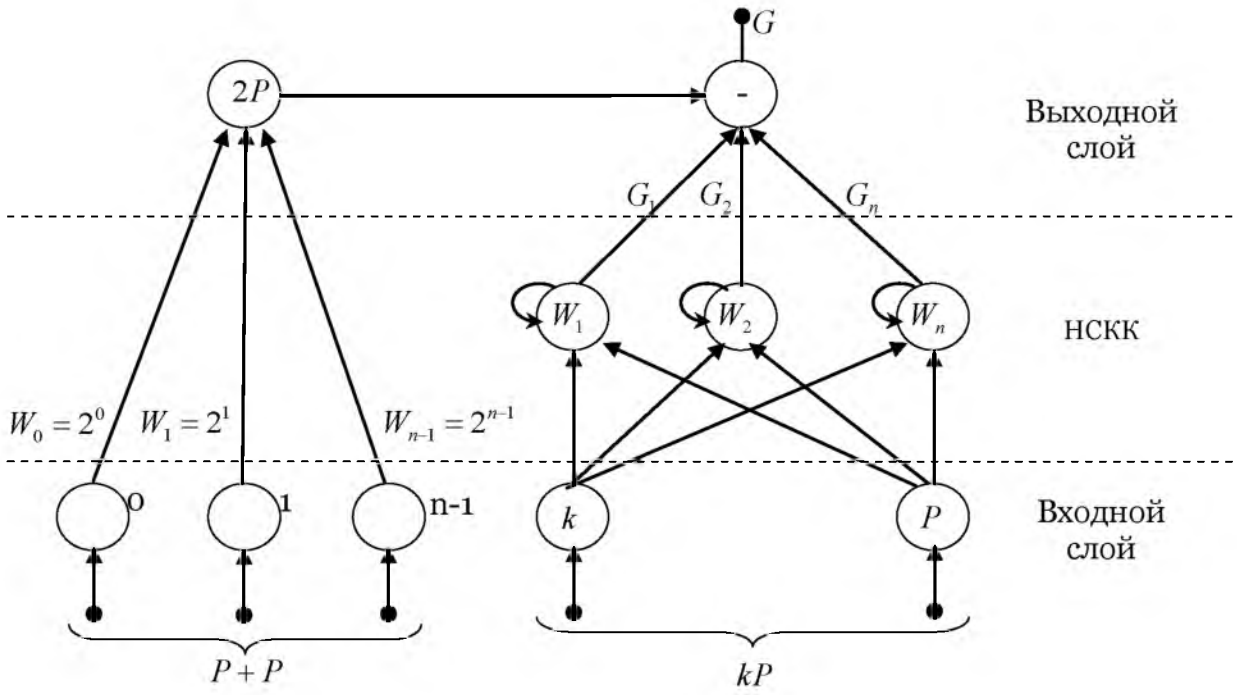

Рис. Архитектура нейронной сети для вычисления kP

Пример 1.

Найти $P+Q$ и $2P$, где $P,Q \in E(F_7)\colon y^2 = x^3 - 3x + 6$ и $P = (1,5)$, $Q = (2,1)$.

Решения

Инициализация НСКК.

Так как в результате произведения двух чисел длиной 3 бита получается число длиной 6 бит, то, для построения НСКК нам необходимо вычислить $W_0 = 1_2$, $W_1 = 10_2$, $W_2 = 100_2$, $W_3 = 1_2$, $W_4 = 10_2$, $W_5 = 100_2$.

1. Случай $P+Q$.

$$k = \frac{y_P - y_Q}{x_P - x_Q} = \frac{5-1}{1-2} = -4 = -100_2$$

$$x_R = k^2 - x_p - x_Q = 10000_2 - 1_2 - 10_2 = (0W_0 + 0W_1 + 0W_2 + 0W_3 + W_4) - 1_2 - 10_2 =$$

$$= 10_2 - 1_2 - 10_2 = -1_2 = 110_2$$

$$y_R = -y_P + k(x_P - x_R) = -1_2 - 100_2(101_2 + 1_2) = -1_2 - 11000_2 = -11001_2 =$$

$$= -(W_0 + 0W_1 + 0W_2 + W_3 + W_4) = -(1_2 + 1_2 + 10_2) = -100_2 = 11_2$$

Следовательно, $R = (x_R, y_R) = P + Q = (6,3)$.

2. Случай $2P$.

$$k = \frac{x_P^2 - 3}{2y_P} = \frac{1_2 - 11_2}{10_2 \cdot 101_2} = \frac{-10_2}{1010_2} = -\frac{1_2}{101_2} = 100_2$$

$$x_R = k^2 - 2x_P = 10000_2 - 10_2 = 0$$

$$y_R = -y_P + k(x_P - x_R) = -101_2 + 100_2(1_2 - 0) = -1_2 = 110_2$$

Следовательно, $R = 2P = (0,6)$.

Из примера видно, что для выполнения модульных операций достаточно произвести один раз инициализацию сети. Умножение и сложение чисел будут выполняться за один раунд нейронной сети.

Заключение

Таким образом, проведенный анализ существующих способов представления точек эллиптической кривой, позволяет сделать следующие выводы:

1. В Якоби-Чудновского сложение двух разных точек эллиптической кривой выполняется за меньшее количество операций, чем в якобиевых координатах и немного быстрее, чем в проективных, однако удвоение точек происходит на одно умножение медленнее, чем в якобиевых.

2. В модифицированных координатах Якоби сложение двух разных точек кривой выполняется за большее количество операций, чем в якобиевых, но при этом удвоение требует на две операции меньше с длинными числами «возведения в квадрат», чем в якобиевых.

3. Удвоение в модифицированных координатах Якоби является наиболее быстрым из всех рассматриваемых координат.

4. Наиболее эффективно можно использовать модифицированные координаты Якоби распараллелив вычисление кратной точки на несколько вычислительных потоков.

5. Из анализа методов вычисления умножения точки на скаляр: одним из самых эффективных является метод NAF, так как он требует меньше памяти компьютера, чем метод окон. Причем с использованием нейронной сети конечного кольца и модифицированного NAF метода получится существенное ускорение выполнения операций скалярного умножения.

Список литературы

1. Cohen H., Miyaji A., Ono T. Efficient elliptic curve exponentiation using mixed coordinates. In: ASIACRYPT'98. Ed. by K. Ohta and D. Pei. Vol. 1514. Lecture Notes in Computer Science. Springer, 1998, pp. 51-65.

2. D. Hankerson, J. Lopez, A. Menezes. Software implementation of elliptic curve cryptography over binary fields. In Cetin K. Koc and C. Paar editors, Workshop and embedded systems (CHES'99) LNCS 1717, pp. 1-24, Springer-Verlag, august 2000.

3. Gordon D. M. A survey of fast exponentiation methods. In: Journal of algorithms 27 (1998), pp. 129-146.

4. Hankerson D., Menezes A., and Vanstone S. Guide to elliptic curve cryptography. Springer Professional Computing Series, Jan. 2003.

5. Joye M. Fast point multiplication on elliptic curves without precomputation. In: WAIFI 2008. Ed. by J. von zur Gathen. J. L. lmana, and C. K. Koc. Vol. 5130. Lecture Notes in Computer Science. Springer, 2008, pp. 36-46.

6. Koyama K., Tsuruoka Y. “Speeding up elliptic cryptosystems by using a signed binary window method”. In: CRYPTO’92. Ed. by E. F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1993, pp. 345-357 (cit. on p. 28).

7. Möller B. “Improved techniques for fast exponentiation”. In: ICISC 2002. Ed. by P. J. Lee and C. H. Lim. Vol. 2587. Lecture Notes in Computer Science. Springer, 2003, pp. 298-312.

8. Muir J.A. Efficient integer representation for cryptographic operations. PhD thesis. University of Waterloo, 2004.

9. Vincent Verneuil Elliptic curve cryptography and security of embedded devices url: http://lfant.math.u-bordeaux1.fr/seminar/slides/2012-06-13-Vincent_Verneuil.pdf

10. Yao A. C.-C. “On the evaluation of powers”. In: SIAM Journal on computing 5.1 (1976), pp. 100-103.

11. Болотов А.А, Гашков С.Б., Фролов А.Б. Элементарное введение в эллиптическую криптографию: Протоколы криптографии на эллиптических кривых. – М.:КомКнига, 2006. – 280 с.

12. Валицкас А. И. Конспект лекций по дисциплине: Элементы абстрактной и компьютерной алгебры. // Тобольск, ТГПИ им. Д. И. Менделеева, 2004.

13. Червяков Н.И., Авербух В.М., Бабенко М.Г., Ляхов П.А., Гладков А.В., Гапочкин А.В. Приближенный метод выполнения немодульных операций в системе остаточных классов// Фундаментальные исследования. 2012. № 6-1. С. 189-193.

14. Червяков Н.И., Бабенко М.Г. Алгебраические подходы к разработке алгоритмов кодирования алфавита точками эллиптической кривой// Нейрокомпьютеры: разработка, применение. 2010. № 9. С. 19-25.

15. Червяков Н.И., Евдокимов А.А., Галушкин А.И., Лавриненко И.Н., Лавриненко А.В. Применение искусственных нейронных сетей и системы остаточных классов в криптографии. – М.: ФИЗМАТЛИТ, 2012. – 280 с.

16. Червяков Н.И., Малофей О. П., Шапошников А. В., Бондарь В.В. Нейронные сети в системах криптографической защиты информации. Нейрокомпьютеры: разработка и применение, 2001, № 10.

17. Червяков Н.И., Сахнюк П.А., Шапошников А.В., Макоха А.Н. Нейрокомпьютеры в остаточных классах – М.: Радиотехника, 2003 – 272 с.

18. Червяков Н.И., Шапошников А.В., Сахнюк П.А. Модель и структура нейронной сети для реализации арифметики системы остаточных классов. – Нейрокомпьютеры: разработка и применение, 2001, № 10, с. 6.

19. Шаханов А.А., Власов А.И., Кузнецов А.С., Поляков Ю.А. Элементная база для реализации параллельных вычислений: тенденции развития. – Труды VII Всероссийской конференции "Нейрокомпьютеры и их применение". – М: ИПРЖ "Радиотехника", 2001, с. 499-531.

DEVELOPMENT OF THE NEURONETWORK METHOD OF SCALAR MULTIPLICATION OF THE POINT OF THE ELLIPTIC CURVE

N. I. CHERVYAKOV
E. S. KIYASHKO

North Caucasian Federal University

e-mail: e.sergeevna.fmf@mail.ru

This paper proposes a new method for calculating the modified scalar multiplication using a neural network of a finite ring. First of all study in this area are aimed at reducing the algorithmic complexity and, at the same time, at increasing of speed. To improve the efficiency is proposed to use the neural network of a finite ring and modified NAF – non-adjacent form method [2].

Key words: elliptic curve, residue number system, neural networks, elliptic curve point multiplication by a scalar, elliptic curve cryptography, projective coordinates.