УДК 681.3.06

И.Д. ГОРБЕНКО, д-р. техн. наук, С.И. ЗБИТНЕВ

РАСШИРЕННОЕ ПОЛЕ ГАЛУА $GF(2^M)$. ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ ПРОСТЕЙШИХ ОПЕРАЦИЙ НАД РАСШИРЕННЫМ ПОЛЕМ $GF(2^M)$

Введение

В последние годы в несимметричной криптографии достаточно широкое распространение нашел математический аппарат теории групп, полей и колец. Так в, ставших уже классическими, криптоалгоритмах класса Эль-Гамаля [1] и в криптопротоколах типа Диффи-Хелмана [2] преобразования осуществляются в простом поле Галуа. В недавно появившихся модификациях этих алгоритмов преобразования осуществляются на эллиптических кривых над простым или расширенным полем Галуа. Применение перечисленных криптоалгоритмов позволяет реализовать ряд преимуществ по сравнению с классическими, к примеру, симметричными криптоалгоритмами. В то же время преобразования больших чисел требуют значительных вычислительных затрат, при этом наибольшей вычислительной сложностью обладает операция умножения по модулю, деление по модулю и возведение в степень по модулю. Рядом преимуществ обладают криптографические преобразования осуществляемые над полем Галуа $GF(2^m)$. Для этого поля также остается актуальной задача вычислительной сложности выполнения базовых операций над большими числами.

Расширенное поле Галуа $GF(2^m)$ имеет два основных базисных представления - полиномиальное и нормальное. Целью настоящей статьи является изучение влияния используемого базиса представления на вычислительную сложность выполнения основных арифметических операций , обоснование и выбора наиболее предпочтительного из них. Определим эти базисы.

При использовании полиномиального базиса каждый элемент поля $GF(2^m)$ представляется двоичным полиномом степени не выше чем $(m-1)$ или битовой строкой его коэффициентов $(a_{m-1}...a_2a_1a_0)$. В наиболее общем случае элемент представляется в виде полинома, имеющего вид

$$b_i=a_{m-1}\,t^{m-1} + \ldots + a_2t^2 + a_1t + a_0 .$$

Для него полиномиальный базис имеет вид

$$B = \{t^{m-1}, \ldots, t^2, t, 1\} .$$

В рассмотренном поле преобразования осуществляются по двойному модулю $p(t),2$, где $p(t)$ является примитивным полиномом. Анализ показывает, что вычислительная сложность выполняемых операций по модулю $p(t)$ зависит от количества ненулевых коэффициентов в примитивном полиноме. Наименьшая вычислительная сложность требуется для трехчлена $p(t)=t^m + t^k + 1$. Однако примитивный трехчлен существует не для всех значений m, для остальных значений m существует пятичлен $p(t)= t^m + t^a + t^b + t^c + 1$, где $m>a>b>c$.

Нормальный базис задается множеством вида

$$b_i =a_0\theta + a_1\theta^2 + a_2\theta^{2^2} + \ldots + a_{m-1}\theta^{2^{m-1}}$$

Наиболее предпочтительным является Гауссовский нормальный базис. Его отличие от нормального базиса заключается в том, что 2^m не делится на 8.

Рассмотрим сущность и алгоритмы основных выполняемых арифметических операций в полиномиальном и нормальном Гауссовском базисе.

1. Алгоритмы операций с полиномиальным базисом

1.1 Сложение по модулю 2

Вход: $a=(a_{m-1}...a_2a_1a_0)$, $b=(b_{m-1}...b_2b_1b_0)$

Выход: $c=(c_{m-1}..c_2c_1c_0)=a+b\ (mod\ 2)$

1. *Для i:=0 до m-1 с шагом 1 выполнить*

1.1 $c_i:=a_i \oplus b_i$

2. *$c=(c_{m-1}...c_2c_1c_0)$, конец*

1.2 Вычитание по модулю 2

Операция вычитания имеет идентичный характер вычисления с операцией сложения.

Вход: $a=(a_{m-1}...a_2a_1a_0)$, $b=(b_{m-1}...b_2b_1b_0)$

Выход: $c=(c_{m-1}...c_2c_1c_0)=a-b(mod\ 2)$

1. *Для i:=0 до m-1 с шагом 1 выполнить*

1.1 $c_i:=a_i \oplus b_i$

2. *$c=(c_{m-1}...c_2c_1c_0)$, конец*

1.3 Приведение *c* по двойному модулю *p(t),2*

Вход: $a=(a_k...a_2a_1a_0)$, $p(t)=(p_m...p_0)$, *где k>m*

Выход: $c=(c_{m-1}...c_2c_1c_0)=a(mod\ p(t),2)$

1. c:=a

2. i:=l(a)-1

3. Если i=m-1 то конец алгоритма

4. Если a_i=0 то перейти на 10

5. j:=m

6. Если j <0 то перейти на 10

7. $c_{i-(m-j)}=c_{i-(m-j)} \oplus p_j$

8. j:=j-1

9. Перейти на 6

10. i:=i-1

11. Перейти на 3

1.4 Умножение по двойному модулю *p(t),2*

Вход: $a=(a_{m-1}...a_2a_1a_0)$, $b=(b_{m-1}...b_2b_1b_0)$, $p(t)=(p_m...p_0)$

Выход: $c=(c_{m-1}...c_2c_1c_0)=a^*\ b(mod\ p(t),2)$

1. c:=0

2. *i:=0*

3. *Если i>l(b)-1 то перейти на 12*

4. *Если b_i=0 то перейти на 10*

5. *j:=0*

6. *Если j>l(a)-1 то перейти на 10*

7. *$c_{i+j}:=c_{i+j} \oplus a_j$*

8. *j:=j+1*

9. *перейти на 6*

10. *i:=i+1*

11. *перейти на 3*

12. *c:=c (mod p(t),2)// по алгоритму 2.3*

13. *$c=(c_{m-1}...c_2c_1c_0)$, конец*

1.5 Возведение в квадрат по двойному модулю *p(t),2*

Вход: $a=(a_{m-1}...a_2a_1a_0)$, $p(t)=(p_m...p_0)$

Выход: $c=a^2(mod\ p(t),2)$

1. c:=0

2. *i:=0*

3. *Если i>l(a)-1 то перейти на 7*

4. $c_{i+i}:=c_{i+i}\oplus a_i$

5. *i:=i+1*

6. *перейти на 3*

7. *c:=c (mod p(t),2)// по алгоритму 2.3*

8. $c=(c_{m-1}\ldots c_2c_1c_0)$, *конец*

1.6 Возведение в квадрат с предвычислениями

Вход: $a=(a_{m-1}\ldots a_2a_1a_0)$, матрица *S // вычисление матрицы описано в* [3]

Выход: $c=a^2$

1. *c:=0*

2. *c:=a*S*

3. $c=(c_{m-1}\ldots c_2c_1c_0)$, *конец*

1.7 Возведение в степень по двойному модулю *p(t),2*

Вход: положительное число *k*, поле $GF(2^m)$ и элемент α.

Выход: $x=\alpha^k(mod\ p(t),2)$.

1.*Пусть* $k=k_r\,k_{r-1}\ldots k_1\,k_0$ *бинарное представление k, где большее количество* k_r *равняется 1.*

2.*Установим* $x:=\alpha$.

3.*Для* $i:=r-1$ *до 0 с шагом -1 выполнить*

3.1 *Установим* $x:=x^2$.

3.2 *Если* $k_i=1$ *то* $x:=\alpha x$.

4.*Выход x, конец.*

Возведение в квадрат и умножение в алгоритме выполняются с помощью 1.4 и 1.5.

1.8 Вычисление квадратного корня по двойному модулю *p(t),2*

Вход: $a=(a_{m-1}\ldots a_2a_1a_0)$, *p(t)*

Выход: $x=\sqrt{\alpha}\ (mod\ p(t),2)$

По [3] $\sqrt{\alpha}=a^{m-1}mod\ p(t)$, поэтому извлечение квадратного корня сводиться к возведению в степень по методике алгоритма 1.7

1.*Пусть* $m-1=k_r\,k_{r-1}\ldots k_1\,k_0$ *бинарное представление k, где большее количество* k_r *равняется 1.*

2.*Установим* $x:=\alpha$.

3.*Для* $i:=r-1$ *до 0 с шагом -1 выполнить*

3.1 *Установим* $x:=x^2$.

3.2 *Если* $k_i=1$ *то* $x:=\alpha x$.

4. *Выход x, конец.*

1.9 Деление по двойному модулю *p(t),2*

Вход: $a=(a_{l-1}\ldots a_2a_1a_0)$, $b=(b_{k-1}\ldots b_2b_1b_0)$, $l<m$, $k<m$

Выход: $c=(c_{m-1}\ldots c_2c_1c_0)=a*b^{-1}(mod\ p(t),2)$

1. *Пусть* $m-1=b_rb_{r-1}\ldots b_0$ *бинарное представление m-1*

2. $\eta:=b$, $k:=1$

3. *Для* $i:=r-1$ *до 0 с шагом 1 выполнить*

3.1 $\mu:=\eta$

3.2 *Для* $j:=1$ *до k с шагом 1 выполнить*

3.2.1 $\mu:=\mu^2$

3.3 $\eta:=\eta*\mu$, $k:=2*k$

3.4 *Если* $b_i=1$, *то* $\eta:=\eta^2*b$, $k:=k+1$

4. *Выход* $a*\eta^2$, *конец.*

Мы рассматриваем общий случай, существуют более эффективные способы для вычисления деления по двойному модулю $p(t),2$.

2. Алгоритмы операций с нормальным базисом

2.1 Сложение

Аналогично алгоритму сложению 1.1 в полиномиальном базисе.

Вход: $a = (a_0\ a_1\ \ldots\ a_{m-1}), b = (b_0\ b_1\ \ldots\ b_{m-1})$

Выход: $c = (c_0\ c_1\ \ldots\ c_{m-1}) = c + b$

1. *Для* $i := m-1$ *до* 0 *с шагом* -1 *выполнить*

1.1 $c_i := a_i \oplus b_i$

2. $c = (c_0\ c_1\ \ldots\ c_{m-1})$, *конец*

2.2 Вычитание

Аналогично алгоритму вычитания 1.2 в полиномиальном базисе.

Вход: $a = (a_0\ a_1\ \ldots\ a_{m-1}), b = (b_0\ b_1\ \ldots\ b_{m-1})$

Выход: $c = (c_0\ c_1\ \ldots\ c_{m-1}) = c + b$

1. *Для* $i := m-1$ *до* 0 *с шагом* -1 *выполнить*

1.1 $c_i := a_i \oplus b_i$

2. $c = (c_0\ c_1\ \ldots\ c_{m-1})$, *конец*

2.3 Возведение в квадрат

Вход: $a = (a_0\ a_1\ \ldots\ a_{m-1})$,

Выход: $c = a^2$

1. $c := a$

2. $c :=$ *ПравыйЦиклическийСдвиг*$(c,1)$

3. $c = (c_{m-1} \ldots c_2 c_1 c_0)$, *конец*

В записи *ПравыйЦиклическийСдвиг*$(c,1)$ выполняется сдвиг на 1 бит.

2.4 Умножение

Вход: $a = (a_0\ a_1\ \ldots\ a_{m-1}), b = (b_0\ b_1\ \ldots\ b_{m-1})$, матрица M, вычисление которой приведено в [3]

Выход: $c = (c_0\ c_1\ \ldots\ c_{m-1})$

1. $x := a$.

2. $y := b$.

3. *Для* $k := 0$ *до* $m-1$ *с шагом* 1 *выполнить*

3.1 *Вычисление произведение при помощи матрицы*

$c_k := x\ M\ y^{tr}$

где y^{tr} *транспонированная матрица вектора* y.

3.2 $x :=$ *ЛевыйЦиклическийСдвиг*$(x,1)$ *и* $y :=$ *ЛевыйЦиклическийСдвиг*$(y,1)$

4. $c = (c_0\ c_1\ \ldots\ c_{m-1})$, *конец*

2.5 Возведение в степень

Алгоритм аналогичен последовательности операций возведения в степень в полиномиальном базисе, но со своими алгоритмами умножения и возведения в степень.

Вход: положительное число k, поле $GF(2^m)$ и элемент α.

Выход: $x = \alpha^k$.

1. *Пусть* $k = k_r\ k_{r-1}\ \ldots\ k_1\ k_0$ *бинарное представление* k, *где большее количество* k_r *равняется 1.*

2. *Установим* $x := \alpha$.

3. *Для* i *от* $r-1$ *до* 0 *с шагом* -1 *выполнить*

3.1 *Установим* $x := x^2$.

3.2 *Если* $k_i = 1$ *то* $x := \alpha x$.

4. *Выход x, конец.*

2.6 Вычисление квадратного корня

Вход: $a = (a_0\ a_1\ \ldots\ a_{m-1})$

Выход: $c = \sqrt{\alpha}$

1. c:=a

2. c:=ЛевыйЦиклическийСдвиг(c,1)

3. $c=(c_{m-1}\ldots c_2 c_1 c_0)$, конец

2.7 Деление

Аналогичен 1.9, но со своими операциями возведение в квадрат и умножения.

Вход: $a = (a_0\ a_1\ \ldots\ a_{m-1}), b = (b_0\ b_1\ \ldots\ b_{m-1})$

Выход: $c=(c_{m-1}\ldots c_2 c_1 c_0)=a*b^{-1}$

4. Пусть $m-1=b_r b_{r-1}\ldots b_0$ бинарное представление m-1

5. $\eta:=b$, $k:=1$

6. For i от r-1 до 0 делать

3.5 $\mu:=\eta$

3.6 For j:=1 до k делать

3.2.1 $\mu:=\mu^2$

3.7 $\eta:=\eta*\mu$, $k:=2*k$

3.8 Если $b_i=1$, то $\eta:=\eta^2*b$, $k:=k+1$

5. Выход $a*\eta^2$

3. Расчет вычислительной сложности

Для оценки вычислительной сложности разобьем команды на классы:

1-я группа – команды сложения, вычитания, пересылки, умножение битов;

2-я группа – команды цикла, перехода, команды сдвигов;

3-я группа – команды умножения и деления.

Используя подходы, изложенные в [4], проведем анализ вычислительной сложности приведенных арифметических операций в полиномиальном и нормальном Гауссовском базисах. Полученные результаты приведены в таблице 1.

Ниже *l(a), l(b), l(m-1)* длина чисел в битах.

Таблица 1

№ алгоритма	№ шага	Среднее количество повторений *j*-го шага в зависимости от входных параметров	Сложность команды
1	2	3	4
1.3	1	*l(a)*	*1*
	2	*1*	*1*
	3	*l(a)-m-1*	*1*
	4	*l(a)-m-1*	*2*
	5	*n(l(a)-m-1)*	*1*
	6	*n(l(a)-m-1)*	*2*
	7	*n(l(a)-m-1)*m*	*1*
	8	*n(l(a)-m-1)*m*	*1*
	9	*n(l(a)-m-1)*m*	*2*
	10	*l(a)-m-1*	*1*
	11	*l(a)-m-1*	*2*

1	2	3	4
1.4	1	*m*	*1*
	2	*1*	*1*
	3	*l(b)*	*2*
	4	*l(b)*	*2*
	5	*l(b)*	*1*
	6	*l(b)*	*2*
	7	*l(b)*l(a)*	*1*
	8	*l(b)*l(a)*	*1*
	9	*l(b)*l(a)*	*2*
	10	*l(b)*	*1*
	11	*l(b)*	*2*
	12	*1*	*Сложн(1.3)*
	13	*1*	*1*
1.5	1	*l(a)*	*1*
	2	*1*	*1*
	3	*l(a)*	*2*
	4	*l(a)*	*1*
	5	*l(a)*	*1*
	6	*l(a)*	*2*
	7	*1*	*Сложн(1.3)*
	8	*1*	*1*
1.6	1	*l(a)*	*1*
	2	*1*	*m(2m-1)*
	3	*1*	*1*
1.7	1	*1*	*0*
	2	*1*	*1*
	3	*l(k)*	*2*
	3.1	*l(k)*	*Сложн(1.6)*
	3.2	*n*l(k)*	*Сложн(1.4)*
	4	*1*	*1*
1.8	1	*1*	*0*
	2	*1*	*1*
	3	*l(m-1)*	*2*
	3.1	*l(m-1)*	*Сложн(1.6)*
	3.2	*n*l(m-1)*	*Сложн(1.4)*
	4	*1*	*1*
1.9	1	*1*	*0*
	2	*l(b)*	*1*
	3	*l(m-1)*	*2*
	3.1	*l(m-1)*	*1*
	3.2	$n*(2^{l(m)+1}+2^{l(m)}-l(m)-2)$	*2*
	3.2.1	$n*(2^{l(m)+1}+2^{l(m)}-l(m)-2)$	*Сложн(1.6)*
	3.3	*l(m-1)*	*Сложн(1.4)*
	3.4	*n* l(m-1)*	*Сложн(1.6)+ Сложн(1.4)*
	4	*1*	*Сложн(1.6) +Сложн(1.4)*

Продолжение таблицы 1

1	2	3	4
2.3	1	$l(a)$	1
	2	1	2
	3	1	1
2.4	1	$l(a)$	1
	2	$l(b)$	1
	3	m	2
	3.1	m	m^2+m
	3.2	m	4
	4	1	1
2.5	1	1	0
	2	1	1
	3	$l(k)$	2
	3.1	$l(k)$	*Сложн(2.3)*
	3.2	$n*l(k)$	*Сложн(2.4)*
	4	1	1
2.6	1	$l(a)$	1
	2	1	2
	3	1	1
2.7	1	1	0
	2	$l(b)$	1
	3	$l(m-1)$	2
	3.1	$l(m-1)$	1
	3.2	$n*(2^{l(m)+1}+2^{l(m)}-l(m)-2)$	2
	3.2.1	$n*(2^{l(m)+1}+2^{l(m)}-l(m)-2)$	*Сложн(2.3)*
	3.3	$l(m-1)$	*Сложн(2.4)*
	3.4	$n*l(m-1)$	*Сложн(2.3)+ Сложн(2.4)*
	4	1	*Сложн(2.3) +Сложн(2.4)*

Заметим, что в алгоритме 2.4 на шаге 2.1 рассчитана сложность через подсчет простейших операций при умножении матрицы размерностью m на m на вектор размерностью m, но при программной реализации эта сложность значительно снизится за счет архитектурно зависимых оптимизаций.

В таблице 2 приведена суммарная вычислительная сложность приведенных алгоритмов, полученная сложением вычислительных сложностей каждого шага соответствующих алгоритмов.

Таблица 2

№ алгоритма	Вычислительная сложность	Примечание
1	2	3
1.3	$l(a)+1+(7.5+2m)(l(a)-m-1)=m^2+3.25m-6.5$	*Для расчета зависимости вычислительной сложности от «m» выбираем l(a) равной 1.5m*
1.4	$m+2+10l(b)+4l(a)l(b)+$ *Сложн(1.3)*$=5m^2+14.25m-4.5$	$l(a)=l(b)=m$
1.5	$7l(a)+2+$ *Сложн(1.3)*$=m^2+10.25m-4.5$	$l(a)=m$
1.6	$l(a)+m(2m-1)+1=2m^2+1$	$l(a)=m$

1	2	3
1.7	$2+l(k)(2+ Сложн(1.6)+0.5 Сложн(1.4))=2+l(k)(4.5\ m^2+7.125m+0.75)$	$l(a)=m$
1.8	$2+l(m-1)(2+ Сложн(1.6)+0.5 Сложн(1.4))=40.5\ m^2+64.125m+8.75$	$n=0.5,\ l(a)=l(b)=m,\ l(m-1)=9,$ так как m ограничено в диапазоне $160 \le m \le 512$ и большая часть чисел представлены девятью битами
1.9	$l(b)+l(m-1)(3+ Сложн(1.4))+ n*(2^{l(m)+1}+ 2^{l(m)}-l(m)-2)(2+ Сложн(1.6))+(Сложн(1.4)+ Сложн(1.6))(0.5l(m-1)+1)=2027.5m^2+207.625m+2883.25$	$n=0.5,\ l(a)=l(b)=m.\ l(m-1)=l(m)=9$
2.3	$l(a)+3=m+3$	$l(a)=m$
2.4	$l(a)+l(b)+m(m^2+m+4)+1=m^3+m^2+8m+1$	$l(a)=l(b)=m$
2.5	$2+l(k)(2+ Сложн(2.3)+0.5 Сложн(2.4))=2+l(k)(0.5\ m^3+0.5m^2+5m+5.5)$	$n=0.5,\ l(a)=m$
2.6	$l(a)+3=m+3$	$l(a)=m$
2.7	$l(b)+l(m-1)(3+ Сложн(2.4))+ n*(2^{l(m)+1}+ 2^{l(m)}-l(m)-2)(2+ Сложн(2.3))+ (Сложн(2.4)+ Сложн(2.3))(0.5l(m-1)+1)=14.5m^3+14.5m^2+1094.5m+4918$	$n=0.5,\ l(a)=l(b)=m,\ l(m-1)=9$

При расчете суммарной сложности все вероятности появления символов 1 и 0 брались равными 50%.

4. Сравнение вычислительной сложности алгоритмов на эллиптической кривой над полем $GF(2^M)$

Проведем сравнение вычислительной сложности основных операций выполняемых при криптографических преобразованиях на эллиптической кривой над полем $GF(2^M)$. При преобразовании на эллиптической кривой основными операциями является: сложение двух точек и удвоение точки. Проведем анализ сложности при выполнении операции удвоение - основной операции при скалярном произведении большого числа на точку, принадлежащей эллиптической кривой.

Удвоение точки вычисляется по формуле $2*(x_1,y_1)=(x_3,y_3)$, где

$$x_3 = \lambda^2 + \lambda + a,$$
$$y_3 = x_1^2 + (\lambda + 1)\, x_3$$
$$\lambda = x_1 + \frac{y_1}{x_1}.$$

1. Полиномиальный базис

$Сложн(\lambda)=2027.5m^2+208.625m+2883.25$

$Сложн(x_3)= 2\ m^2+2m+1$

$Сложн(y_3)= 7\ m^2+16.25m-3.5$

Сложн(Суммарная1)$= Сложн(\lambda)+Сложн(x_3)+ Сложн(y_3)= 2036.5m^2 + 226.875m + 2880.75$

2. Нормальный базис

$Сложн(\lambda)=14.5m^3+14.5m^2+1095.5m+4918$

$Сложн(x_3)=3m+3$

Сложн(y_3) = $m^3+m^2+11m+4$

Сложн(Суммарная2)= *Сложн(λ)* + *Сложн(x_3)* + *Сложн(y_3)* = $15.5m^3 + 15.5m^2 + 1109.5m + 4925$

Теперь мы можем проанализировать зависимость вычислительной сложности алгоритмов от значения m. Полученные результаты приведены в таблице 3.

Таблица 3

Значение m	**Полиномиальный базис**	**Нормальный базис**
1	2	3
15	14515	2417
30	57579	14708
45	129281	46833
60	229621	108602
75	358600	209824
90	516217	360307
105	702472	569859
120	917365	848289
135	1160897	1205406
150	1433067	1651018
165	1733875	2194934
180	2063322	2846963
195	2421407	3616913
256	4172657	8167237
512	16686728	65156593
1024	66739382	520637257

Полученные результаты представлены на рисунке 1.

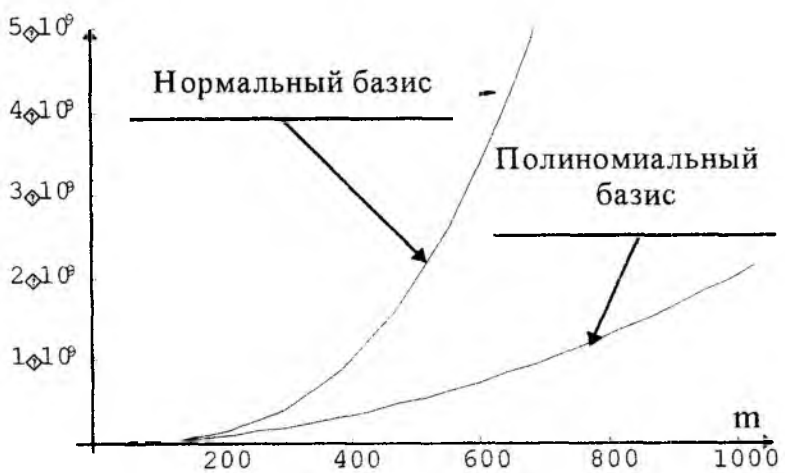

Рис. 1

Кривые, показанные на рисунке 1, имеют две точки пересечения: $m \approx 1.254$, $m \approx 129.941$. Но масштаб рисунка 1 не дает возможность этого увидеть, поэтому приведем рисунок 2 с более крупным масштабом.

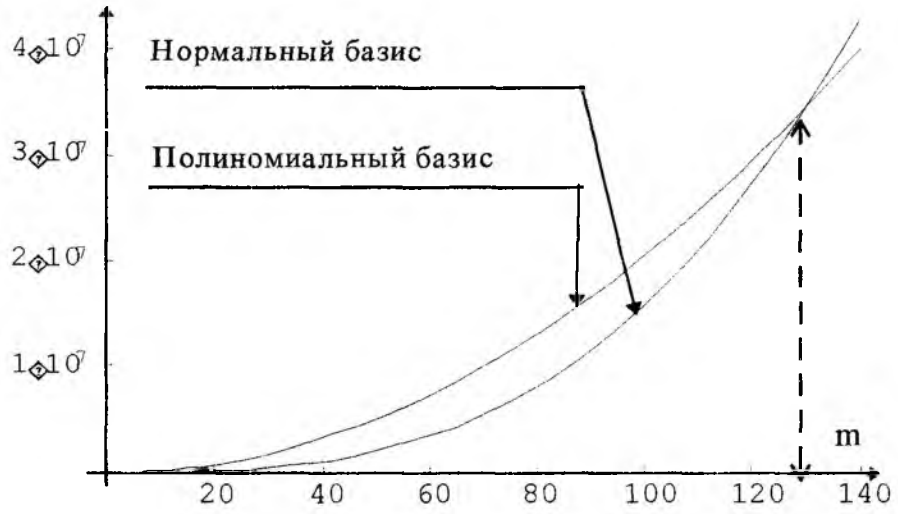

Рис. 2

Заключение

Таким образом, более предпочтительным является нормальный Гауссовский базис, сточки зрения минимизации вычислительной сложности при $m<129$, а при $m>129$ - полиномиальный базис. Учитывая то, что преобразования на эллиптической кривой используют $m>160$, в целом более предпочтительным является полиномиальный базис.

Список литературы: **1.** T. *ELGAMAL*, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory, 31 (1985), 469-472. **2.** *M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, AND M. Wiener*, Minimal key lengths for symmetric ciphers to provide adequate commercial security, January 1996. **3.** *IEEE P1363* / D11 (Draft Version 11). Standard Specifications for Public Key Cryptography. Annex A (Informative). Number-Theoretic Background. **4.** *Кнут Д.* Основы программирования для ЭВМ: В 3 т. – М.: Мир, 1977. Т.2. Получисленные алгоритмы.– 724 с.

Харьковский государственный технический университет радиоэлектроники

Поступила в редколлегию 15.03.2000