Проблемы эксплуатации и защиты ИКС-2014

Модифицированный метод инвертирования в двоичном поле

Национальный авиационный университет

Кафедра БИТ

Соискатель: Мария Булах

Научный руководитель: к.т.н. Влад Ковтун

Содержание

- Введение
- Актуальность
- Известные алгоритмы
- Расширенный алгоритм Эвклида
- Модифицированный расширенный алгоритм Эвклида
- Сравнение производительности
- Выводы

Введение

Цель: повышение производительности операции мультипликативного инвертирования в двоичном поле посредством алгоритмов «следующего подходящего индекса» и «значимых слагаемых».

Объект: процесс арифметических преобразований полиномов в двоичном поле.

Предмет: операция мультипликативного инвертирования полиномов в двоичном поле.

Актуальность

Соответствие сложности операций: **S~0.12M** и **I~11M**

Повышение производительности

- Увеличение разрядности машинных слов
- Использование специализированных потоковых команд процессоров (MMX, SSE, SSE2, SSE3, SSE4, SSE5)
- Распараллеливание (многопоточность)
- **Совершенствование алгоритмов (deg, shl, add) оперирующих полиномами**
- Совершенствование структур данных представления полиномов

Известные алгоритмы

- На основе Малой теоремы Ферма (exp)
- Itoh-Tsujii (exp)
- Матричный полином
- Различные модификации Itoh-Tsujii (exp)
- **Инвертирование на основе расширенного алгоритма Эвклида**
- Почти инвертирования (Almost Inversion)
- Модифицированное почти инвертирование (Modified Almost Inversion)

Известный алгоритм

РАСШИРЕННЫЙ АЛГОРИТМ ЭВКЛИДА

Расширенный алгоритм Эвклида

$a \in GF\left(2^m\right)$ - ненулевые элементы в поле GF(2^m);

$ba + df = u$ и $ca + ef = v$ - инварианты, на которых основан алгоритм;

$f(x)$ - неприводимый полином в поле GF(2^m);

d и e - полиномы, которые вычисляются неявно;

u и v - полиномы, степень которых уменьшается;

b и c - полиномы, степень которых растёт.

Расширенный алгоритм Эвклида

Алгоритм 1. Расширенный алгоритм Эвклида для мультипликативного инвертирования в поле $GF(2^m)$.

Вход: элемент $a \in GF(2^m),\ a \neq 0$.

Выход: $a^{-1} \bmod f(x)$.

1. $b \leftarrow 1,\ c \leftarrow 0,\ u \leftarrow a,\ v \leftarrow f$.

2. While $\deg(u) \neq 0$ do

2.1. $j \leftarrow \deg(u) - \deg(v)$.

2.2. if $j < 0$ then $u \leftrightarrow v,\ b \leftrightarrow c,\ j \leftarrow -j$.

2.3. $u \leftarrow u + x^j v,\ b \leftarrow b + x^j c$.

3. Return (b).

РАЭ: Сложность

$$I_{avr}(A_1) = k(2I_{\deg} + 2I_{add} + 2I_{shl}) + 2k/3(2I_{swp})$$

$I_{\deg}$ - сложность алгоритма вычисления степени полинома;

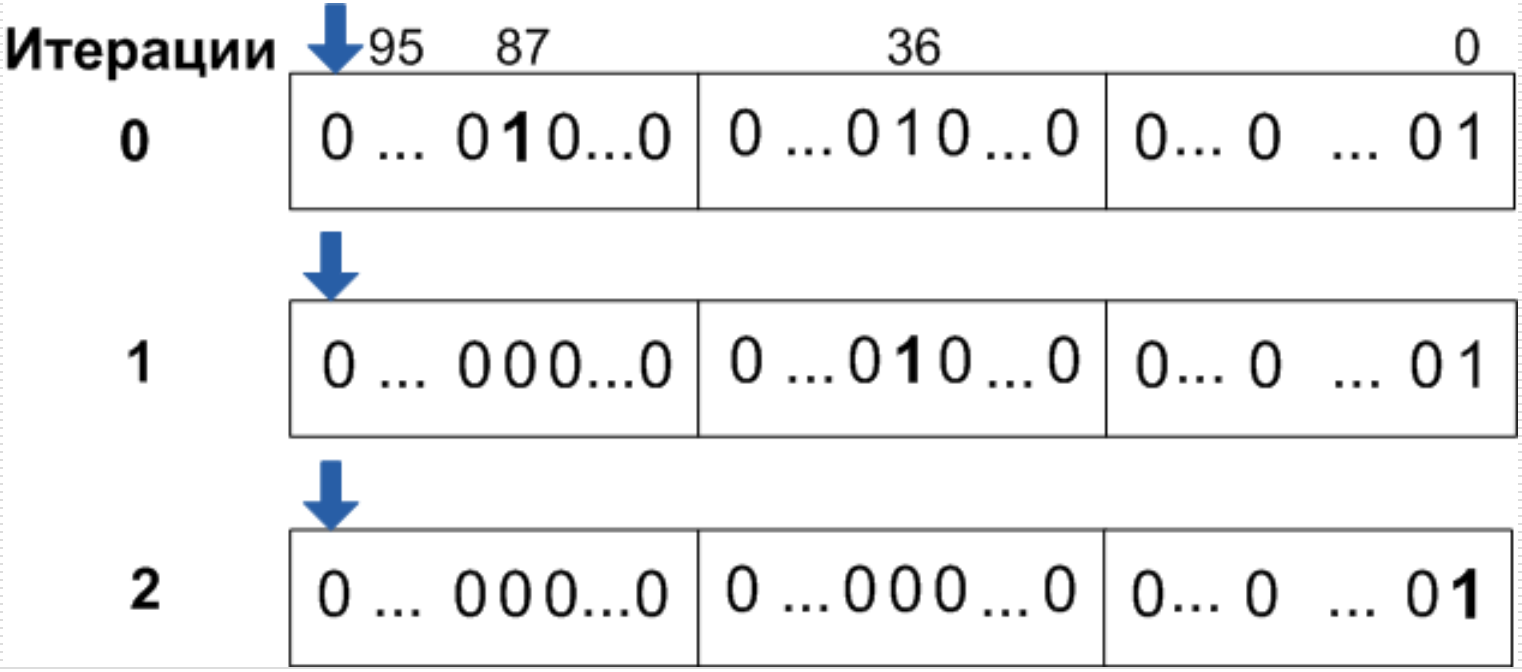
I_{add} - сложность алгоритма сложения двух полиномов;

I_{shl} - сложность алгоритма сдвига на произвольное число бит (может превышать длину машинного слова);

I_{swp} - сложность алгоритма обмена двух полиномов.

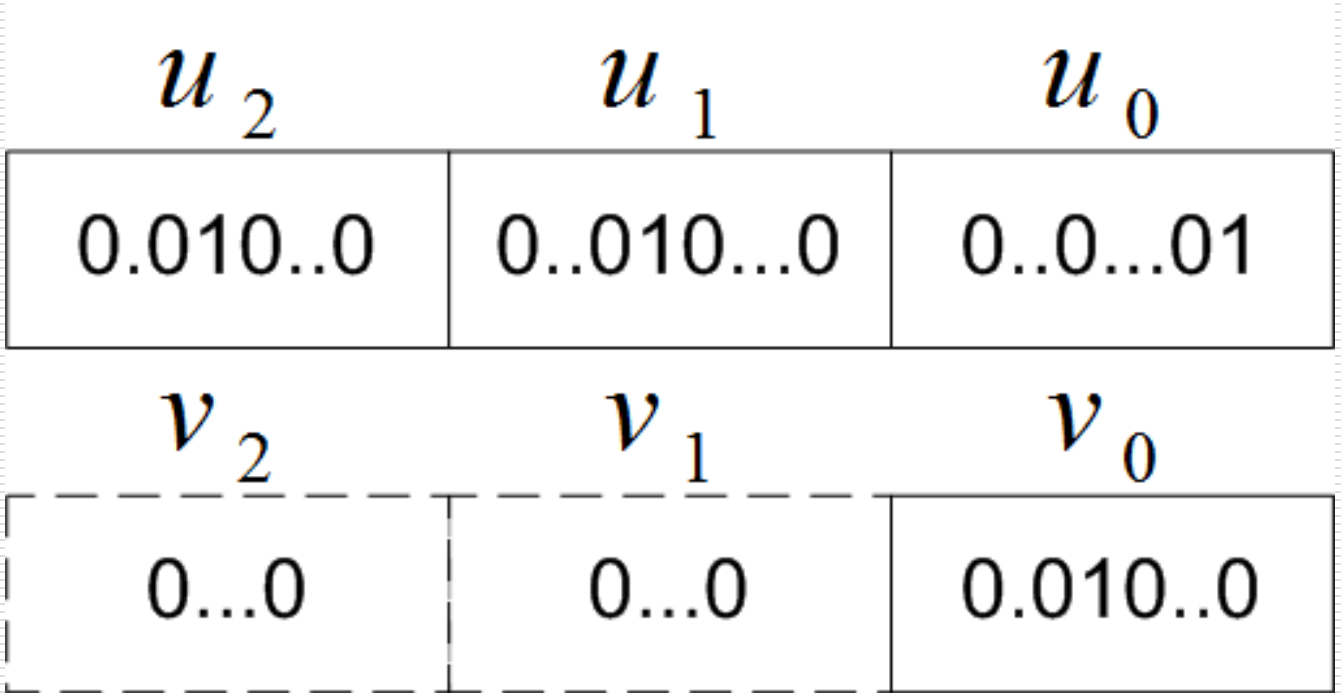
РАЭ: Степень полинома

$$u(x) = x^{87} + x^{36} + 1$$



РАЭ: Сдвиг полинома (до)

$$u(x) = x^{87} + x^{36} + 1,\ v(x) = x^{25}$$



РАЭ: Сдвиг полинома (после)

$$u(x) = x^{87} + x^{36} + 1,\ v(x) = x^{87}$$

u_2	u_1	u_0
0.010..0	0..010...0	0..0...01

v_2	v_1	v_0
0.010..0	0...0	0...0

РАЭ: Сложение полиномов

$$u(x) = x^{87} + x^{36} + 1,\ v(x) = x^{87},\ u(x) \oplus v(x) = x^{36} + 1$$

u_2	u_1	u_0
0.010..0	0..010...0	0..0...01
$\oplus$ v_2	v_1	v_0
0.010..0	0...0	0...0
$u_2 \oplus v_2$	$u_1 \oplus v_1$	$u_0 \oplus v_0$

РАЭ: Пример

Итерация	*deg(u)*	*deg(v)*	*deg(b)*	*deg(c)*
0	87	89	0	-1
1	88	87	2	0
2	86	87	2	0
3	86	86	3	2
4	84	86	3	2
5	84	84	5	3
6	83	84	5	3
7	81	83	6	5
8	82	81	8	6
⋮				
76	7	8	81	80
77	7	7	82	81
78	6	7	82	81
79	4	6	83	82
80	4	4	85	83
81	1	4	85	83
82	1	1	88	85

$$ba + df = u$$

$$ca + ef = v$$

$$a(x) = x^{87}+x^{86}+x^{85}+x^{82}+x^{81}+x^{80}+x^{78}+x^{77}+x^{76}+x^{75}+x^{74}+x^{71}+x^{70}+x^{69}+x^{64}+$$

$$+x^{28}+x^{25}+x^{18}+x^{17}+x^{15}+x^{12}+x^{11}+x^{8}+x^{7}+x^{6}+x^{5}+x^{4}+1$$

*-степень полиномов *u* и *v* уменьшается, а степень полиномов *b* и *c* - растет

РАЭ: Недостатки

- Степень ***v(x)*** изначально определена и равна степени ***m*** неприводимого полинома ***f(x)*** либо равна степени ***u(x)***. Производится избыточное вычисление ***deg(v)***.
- Поиск степени ***u(x)*** и ***v(x)*** на каждой итерации осуществляется с самого старшего машинного слова. Производятся избыточные проверки.
- При сдвиге ***v(x)*** *и* ***c(x)***, с последующим сложением с ***u(x)*** и ***b(x)*** производятся операции со всеми машинными словами, даже с заведомо равными 0.

Предложенный алгоритм

МОДИФИЦИРОВАННЫЙ РАСШИРЕННЫЙ АЛГОРИТМ ЭВКЛИДА

МРАЭ: РАЭ+Усовершенствования

- Алгоритм «следующего подходящего»:

  Степень ***u(x)*** уменьшается хотя бы на 1, поэтому незачем вычислять ***deg(u)*** на каждой итерации, нужно лишь уточнять ее, основываясь на предыдущих значениях. Количество проверок уменьшается в 2 раза.

- Алгоритм «учет значимых элементов»:

  Степень ***u(x)*** и ***v(x)*** постоянно уменьшается, а степень ***b(x)*** и ***c(x)*** – растёт. Это дает возможность выполнять сдвиг и сложение не всех машинных слов, а лишь значимых (отличных от нуля). Количество сдвигов и сложений машинных слов уменьшается в 2 раза.

МРАЭ: РАЭ+Усовершенствования

- Алгоритм-трюк «вычисления номера старшего значащего бита в машинном слове»:

  Используется известный алгоритм вычисления номера старшего значащего бита в машинном слове без операций ветвления, что позволяет максимально эффективно использовать возможности современных суперскалярных (конвееризации команд) процессоров.

МРАЭ: РАЭ+Усовершенствования

Вход: $a \in \mathbf{GF}(2^m)$; $n = \lceil \frac{m}{w} \rceil$, где n-число машинных слов, которое занимает полином; w-ширина машинного слова, обычно $w = 32$.

Выход: $\deg(a)$.

1. $i \leftarrow n-1$.

2. While $\left(a_i^{(32)} \neq 0\right) \&\& (i > 0)$

2.1. $i \leftarrow i-1$.

3. $t \leftarrow a_i^{(32)}$.

4. $t \leftarrow t \mid (t >> 1)$, $t \leftarrow t \mid (t >> 2)$, $t \leftarrow t \mid (t >> 4)$, $t \leftarrow t \mid (t >> 8)$, $t \leftarrow t \mid (t >> 16)$.

5. $t \leftarrow t - ((t >> 1) \& 0\text{x}55555555)$, $t \leftarrow (t \& 0\text{x}33333333) + ((t >> 2) \& 0\text{x}33333333)$,
$t \leftarrow ((t + (t >> 4)) \& 0\text{xf0f0f0f}) \cdot 0\text{x}1010101) >> 24$.

6. Return $((i << 5) + t - 1)$.

Модифицированный РАЭ

Алгоритм 3. Модифицированный расширенный алгоритм Эвклида для мультипликативного инвертирования в поле $\mathbf{GF}(2^m)$.

Вход: $a \in \mathbf{GF}(2^m)$, $a \neq 0$, $n = \lceil \frac{m}{w} \rceil$, где n-число машинных слов, которое занимает полином; w-ширина машинного слова, обычно $w = 32$.

Выход: $a^{-1} \bmod f(x)$.

1. $b \leftarrow 1$, $c \leftarrow 0$, $u \leftarrow a$, $v \leftarrow f$.
2. $i \leftarrow n-1$.
3. While $(u_i^{(32)} \neq 0) \&\& (i > 0)$
3.1. $i \leftarrow i-1$.
4. $t \leftarrow u_i^{(32)}$.
5. $t \leftarrow t \mid (t >> 1)$, $t \leftarrow t \mid (t >> 2)$, $t \leftarrow t \mid (t >> 4)$, $t \leftarrow t \mid (t >> 8)$, $t \leftarrow t \mid (t >> 16)$.
6. $t \leftarrow t - ((t >> 1) \& 0x55555555)$, $t \leftarrow (t \& 0x33333333) + ((t >> 2) \& 0x33333333)$, $t \leftarrow ((t + (t >> 4) \& 0xf0f0f0f) \cdot 0x1010101) >> 24$.
7. $\deg U = ((i << 5) + t - 1)$.
8. $\deg V = m$.

Модифицированный РАЭ

9. While $(\deg U > 0)$ do

9.1. If $(\deg U < \deg V)$ then $k \leftarrow \deg V - \deg U$, $u \leftrightarrow v$. $b \leftrightarrow c$.

9.2. else $k \leftarrow \deg U - \deg V$.

9.3. if $(k > 0)$ then $u = u + x^k \cdot v$, $b = b + x^k \cdot c$.

9.4. else $u = u + v$, $b = b + c$.

9.5. If $(\deg U < \deg V)$ then $\deg V = \deg U$.

9.6. While $\left(u_i^{(32)} \neq 0\right) \&\& (i > 0)$

9.6.1. $i \leftarrow i - 1$.

9.7. $t \leftarrow u_i^{(32)}$.

9.8. $t \leftarrow t \,|\, (t >> 1)$, $t \leftarrow t \,|\, (t >> 2)$, $t \leftarrow t \,|\, (t >> 4)$, $t \leftarrow t \,|\, (t >> 8)$, $t \leftarrow t \,|\, (t >> 16)$.

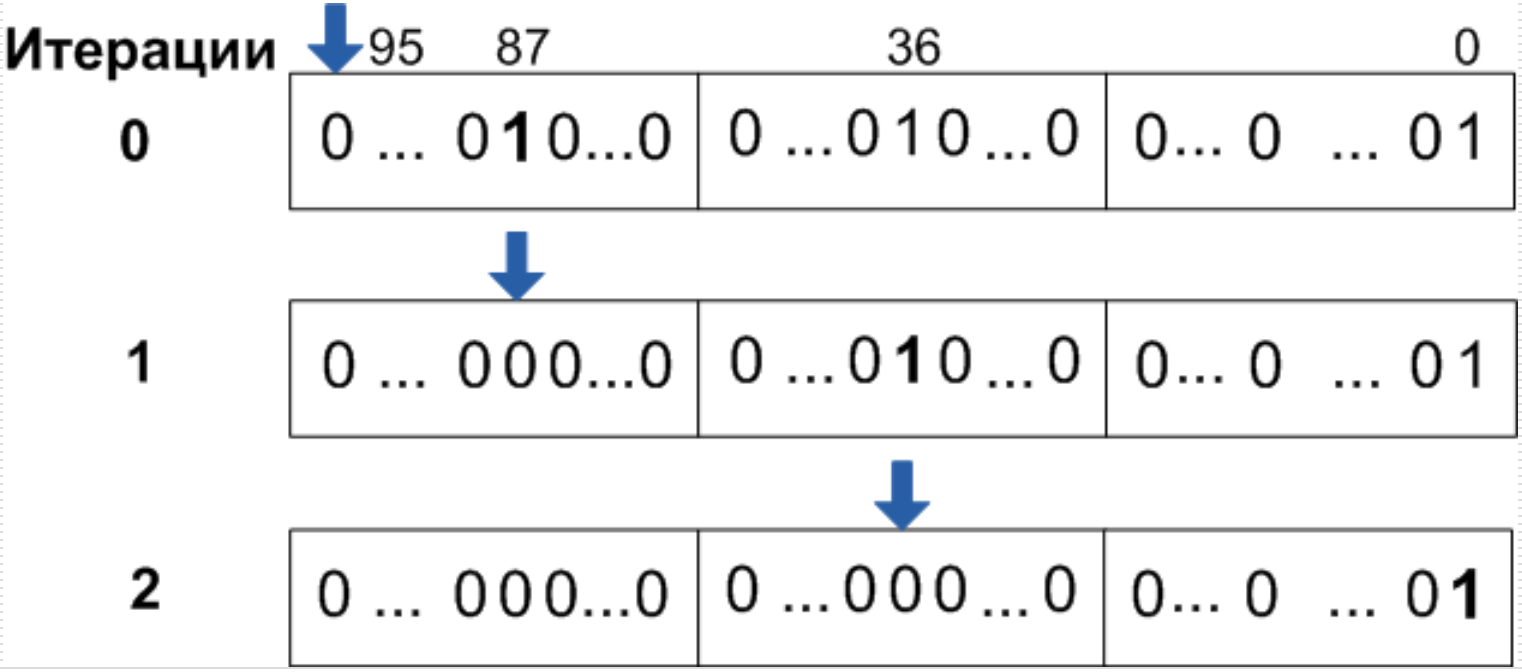
9.9. $t \leftarrow t - ((t >> 1) \& 0\text{x}55555555)$, $t \leftarrow (t \& 0\text{x}33333333) + ((t >> 2) \& 0\text{x}33333333)$,
$t \leftarrow ((t + (t >> 4) \& 0\text{xf0f0f0f}) \cdot 0\text{x}1010101) >> 24$.

9.10. $\deg U = ((i << 5) + t - 1)$.

10. Return (b).

МРАЭ: Степень полинома – следующий подходящий индекс

$$u(x) = x^{87} + x^{36} + 1$$



МРАЭ: Сдвиг полинома - значимые слагаемые (до)

$$u(x) = x^{87} + x^{36} + 1,\ v(x) = x^{25}$$

u_2	u_1	u_0
0.010..0	0..010...0	0..0...01

v_2	v_1	v_0
0...0	0...0	0.010..0

МРАЭ: Сдвиг полинома - значимые слагаемые (после)

$$u(x) = x^{87} + x^{36} + 1,\ v(x) = x^{87}$$

u_2	u_1	u_0
0.010..0	0..010...0	0..0...01

v_2	v_1	v_0
0.010..0	0...0	0...0

МРАЭ: Сложение полиномов - значимые слагаемые

$$u(x) = x^{87} + x^{36} + 1,\ v(x) = x^{87},\ u(x) \oplus v(x) = x^{36} + 1$$

	u_2	u_1	u_0
	0.010..0	0..010...0	0..0...01
$\oplus$	v_2	v_1	v_0
	0.010..0	0...0	0...0
	$u_2 \oplus v_2$	u_1	u_0

МРАЭ: Сложность

$$I_{avr}(A_3) = k\left(I_{\deg} + \tfrac{1}{2}\left(2I_{add} + 2I_{shl}\right)\right) + 2k/3\left(2I_{swp}\right)$$

$I_{\deg}$ - сложность алгоритма вычисления степени полинома,

I_{add} - сложность алгоритма сложения двух полиномов,

I_{shl} - сложность алгоритма сдвига на произвольное число бит (может превышать длину машинного слова),

I_{swp} - сложность алгоритма обмена двух полиномов.

МРАЭ vs РАЭ

СРАВНЕНИЕ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ

Сравнение сложности

$$I_{avr}(A_1) = k(2I_{\deg} + 2I_{add} + 2I_{shl}) + 2k/3(2I_{swp})$$

$$I_{avr}(A_3) = k(I_{\deg} + \tfrac{1}{2}(2I_{add} + 2I_{shl})) + 2k/3(2I_{swp})$$

МРАЭ vs РАЭ

СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ

Программная реализация

- ОС: Microsoft Windows 7 Pro x86-64
- Компиляторы (Target-x86):
  - MCC 2010 - Microsoft Visual C++ 2010 (/O3, поддержка SSE2)
  - ICC XE 2013 - Intel C++ Compiler XE2013 (/O3, поддержка SSE4.2)

Эксперимент: Условия

- Среднее для 1 млн. операций
- Поля рекомендованные:
  - ДСТУ 4145-2002
  - FIPS 186-3
- Вычислительные системы:
  - Intel Core i3-350M (notebook)
  - Intel Core i5-3570 (desktop)
  - Intel Core i5-4670 (desktop)

Тестовые поля

m	Неприводимый полином
89	$x^{89}+x^{38}+1$
163	$x^{163}+x^{7}+x^{6}+x^{3}+1$
167	$x^{167}+x^{6}+1$
173	$x^{173}+x^{10}+x^{2}+x+1$
179	$x^{179}+x^{4}+x^{2}+x+1$
191	$x^{191}+x^{9}+1$
233	$x^{233}+x^{9}+x^{4}+x+1$
233	$x^{233}+x^{74}+1$
257	$x^{257}+x^{12}+1$
283	$x^{283}+x^{12}+x^{7}+x^{5}+1$
307	$x^{307}+x^{8}+x^{4}+x^{2}+1$
367	$x^{367}+x^{21}+1$
409	$x^{409}+x^{87}+1$
431	$x^{431}+x^{5}+x^{3}+x+1$
571	$x^{571}+x^{10}+x^{5}+x^{2}+1$

Производительность

m	Время, мкс											
	Intel Core i3-350M				Intel Core i5-3570				Intel Core i5-4670			
	ICC XE2013		MCC2010		ICC XE2013		MCC2010		ICC XE2013		MCC2010	
	Inv	Inv*	Inv	Inv*	Inv	Inv*	Inv	Inv*	Inv	Inv*	Inv	Inv*
89	6,71	5,44	6,27	5,10	2,53	1,95	2,76	1,84	2,53	1,95	2,37	1,84
163	16,08	11,34	14,38	11,96	6,85	4,05	6,33	4,65	6,85	3,95	6,13	4,05
191	18,17	14,52	17,38	14,71	7,73	5,46	7,85	5,48	7,73	5,26	7,65	5,48
233	27,13	22,12	24,57	19,39	11,80	7,04	11,59	7,65	11,80	7,02	11,02	6,90
257	31,56	25,18	27,93	24,54	13,21	7,99	13,33	8,56	12,17	7,81	12,33	7,60
307	37,72	30,45	34,24	26,11	17,98	11,11	17,64	12,41	17,68	9,58	17,54	11,41
367	53,18	42,17	46,05	33,81	23,35	14,78	21,84	16,63	22,35	13,18	21,64	14,63
409	61,31	40,18	54,48	47,76	26,41	16,97	26,81	18,51	25,97	14,93	26,41	17,51
431	67,75	54,24	59,10	44,03	28,99	17,86	29,29	19,25	28,27	17,00	28,99	18,25
571	103,44	64,86	94,42	70,26	46,46	25,47	44,87	26,98	43,39	24,64	44,67	26,83

* – MPAЭ.

МРАЭ vs РАЭ

ВЫВОДЫ

Выводы

- Глубокий анализ РАЭ позволил выявить потенциальные улучшения:
  - Можно отказаться от вычисления степени ***v(x)***, она либо не изменяется либо равна степени ***u(x)***.
  - Можно воспользоваться алгоритмом «следующего подходящего», для вычисления степени ***u(x)***, т.е. производить последовательное уточнение степени, а не вычисление с «0».
  - Можно воспользоваться «трюком», для вычисления старшего отличного бита в машинном слове, что позволило отказаться от операций сравнения и перехода.

Выводы

- Выявленные улучшения РАЭ были реализованы в МРАЭ.
- Вычислительная сложность МРАЭ меньше 2 раза, чем в РАЭ.
- Программная реализация МРАЭ, в среднем, обладает большей производительностью чем РАЭ, на 15-50%, с ростом размера поля.

Выводы

- Предложенная программная реализация МРАЭ не ориентирована на многопоточное выполнение, что не позволяет полностью реализовать потенциал современных многоядерных процессоров.
- Применение МРАЭ, в программной реализации алгоритмов формирования и проверки ЭЦП согласно ДСТУ 4145-2002, позволило увеличить производительность:
  - на 16-50% в аффинном представлении точек
  - на 2-4% в проективном представлении Лопеса-Дахаба.

Дальнейшие исследования

- Развитие современных микропроцессоров направлено на увеличение числа потоков выполнения команд, что в свою очередь требует разрабатывать полноценные алгоритмы для эффективной программной реализации на перспективных микропроцессорах.
- NVIDIA, уже сейчас предлагает графические процессоры с числом ядер более 512, а также удобную среду разработки CUDA, для создания полноценных многопоточных приложений.
- Дальнейшие исследования будут направлены на изучение и эффективное распараллеливание алгоритмов арифметических операций в поле **GF**(2^m).

Вопросы?

Спасибо за внимание!

Национальный авиационный университет

Кафедра безопасности информационных технологий

Мария Булах

email: bulakh.masha@gmail.com