The Problems of Operation and protection Information & Communication Systems-2014

The Modified Method of Inversion in Binary Field

National Aviation University
Chair Information Security Systems
Applicant: Mary Bulakh
Advisor: Ph.D. Vlad Kovtun

Content

- Introduction
- Actuality
- Well-known Algorithms
- Extended Euclidean Algorithm
- Modified Extended Euclidean Algorithm
- Performance comparison
- Conclusion

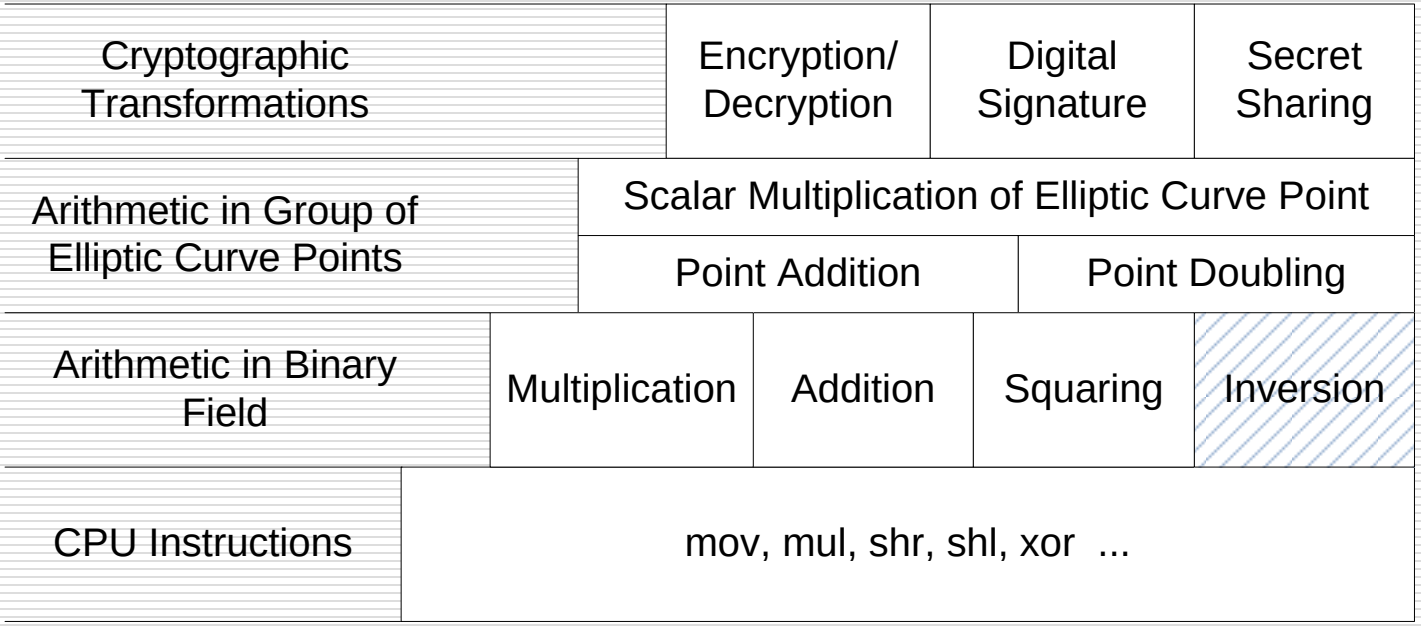
Introduction

Objective: performance improvement of multiplicative inversion operation in binary field by algorithms "next fit index" and "significant terms".

Object: process of arithmetic operations over polynomials in binary field.

Subject: multiplicative inversion operation over polynomials in binary field.

Actuality



Operations complexity: **S~0.12M** and **I~11M**

Efficiency improving

- Increasing machine words length
- Using specialized streaming command of CPU (MMX, SSE, SSE2, SSE3, SSE4, SSE5)
- Parallelizing (multithreading)
- **Improving of algorithms (deg, shl, add) for polynomials manipulating**
- Improvement of data structures for polynomial representation

Acquainted algorithm

- Algorithm based on Fermat's Little Theorem
- Itoh-Tsujii (exp)
- Matrix polynomial
- Various modifications Itoh-Tsujii (exp)
- **Inversion based on the Extended Euclidean Algorithm**
- Almost Inversion
- Modified Almost Inversion

Well-known Algorithm

EXTENDED EUCLIDEAN ALGORITHM

Extended Euclidean Algorithm

$a \in \mathbf{GF}(2^m)$ - non-zero element of $\mathbf{GF}(2^m)$;

$ba + df = u$ and $ca + ef = v$ - invariants, what lies in base of inversion algorithm;

$f(x)$ - irreducible polynomial;

d and e - implicitly compute polynomials;

u and v - polynomials with permanently decreasing degree;

b and c - polynomials with permanently increasing degree.

$b = a^{-1} \bmod f(x).$

Extended Euclidean Algorithm

Algorithm 1. Extended Euclidean Algorithm for multiplicative inversion in $\mathbf{GF}(2^m)$.

Input: $a \in \mathbf{GF}(2^m)$, $a \neq 0$.

Output: $a^{-1} \bmod f(x)$.

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).

EEA: Complexity

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

$I_{\deg}$ - complexity of algorithm for computing polynomial degree;

I_{add} - complexity of algorithm for adding two polynomials;

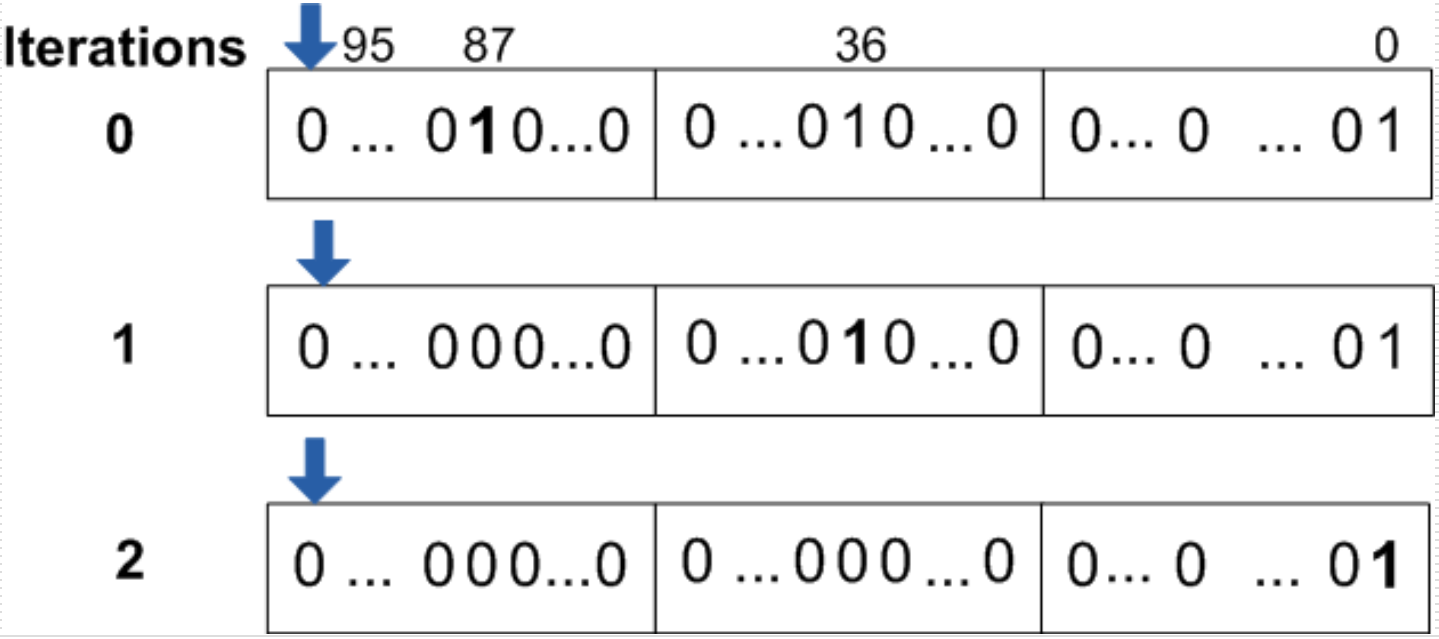
I_{shl} - complexity of algorithm for shifting to arbitrary number of bits (may to exceed the length of machine word);

I_{swp} - complexity of algorithm for exchange two polynomials.

* k- number of iterations.

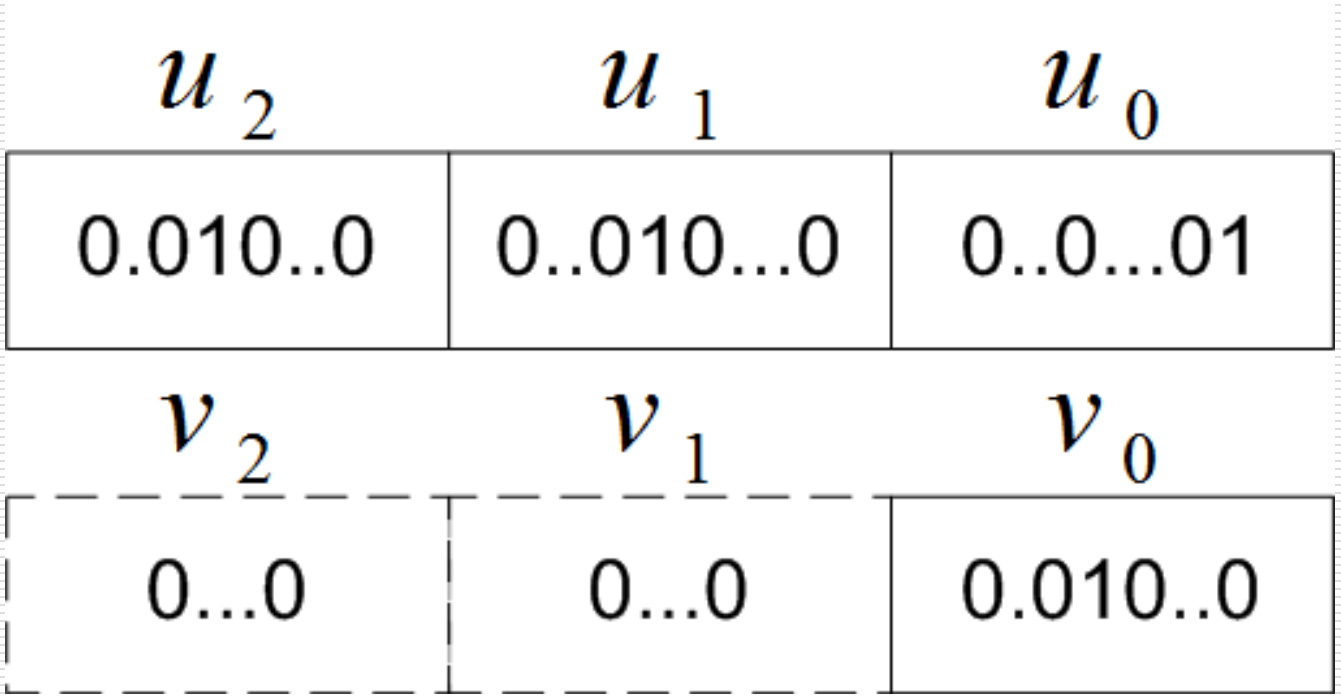
EEA: Degree of polynomial

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



EEA: Shift of polynomial (before)

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



EEA: Shift of polynomial (after)

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

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

EEA: Addition of polynomials

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

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
$u_2 \oplus v_2$	$u_1 \oplus v_1$	$u_0 \oplus v_0$

EEA: Exemple

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

$$ba + df = u$$

$$ca + ef = v$$

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

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

* - degree of polynomials ***u*** and ***v*** permanently decreasing, and degree of polynomials ***b*** and ***c*** is increasing

EEA: Limitations

- Degree of ***v(x)*** is initially defined and equal ***m*** - degree of irreducible polynomial ***f(x)*** or equal degree ***u(x)***. Redundant computation ***deg (v).***
- Search degree of ***u(x)*** and ***v(x)*** performed at each iteration from most significant machine word. Redundant checks on each iterations..
- While shifts ***v(x)*** and ***c(x)***, with following additions ***u(x)*** and ***b(x)*** are performed with all machine words (even obvious equal to zero).

Proposed Algorithm

MODIFIED EXTENDED EUCLIDEAN ALGORITHM

MEEA: EEA+Improvements

- Algorithm «next fit index»:

  Degree of ***u(x)*** decreases by at least 1, so there is no need to compute ***deg(u)*** at each iteration, just need to refine previous values. The number of checks decreases in 2 times.

- Algorithm «significant terms»:

  Degree of polynomials ***u(x)*** and ***v(x)*** permanently decreasing, and degree of polynomials ***b(x)*** and ***c(x)*** is increasing. This allows shifting and adding not all machine words, but only significant (non-zero). Number of additions and shifts decreased almost in 2 times.

MEEA: EEA+Improvements

- Algorithm-trick «computing MSB in machine word»:

  used algorithm for computing the number of MSB in machine word without branching commands. This allows useing a best opportunities (pipelining commands) of modern superscalar CPU.

MEEA: EEA+Improvements

Algorithm 2. «Trick» - calculation of the polynomial degree.

Input: $a \in \mathbf{GF}(2^m)$; $n = \lceil \frac{m}{w} \rceil$, n-number of machine words, which takes polynomial; w-wide of machine word, usually $w = 32$.

Output: $\deg(a)$.

1. $i \leftarrow n-1$.
2. While $(a_i^{(32)} \neq 0) \&\& (i > 0)$
2.1. $i \leftarrow i-1$.
3. $t \leftarrow a_i^{(32)}$.
4. $t \leftarrow t \,|\, (t >> 1)$, $t \leftarrow t \,|\, (t >> 2)$, $t \leftarrow t \,|\, (t >> 4)$, $t \leftarrow t \,|\, (t >> 8)$, $t \leftarrow t \,|\, (t >> 16)$.
5. $t \leftarrow t - ((t >> 1) \& 0x55555555)$, $t \leftarrow (t \& 0x33333333) + ((t >> 2) \& 0x33333333)$,
$t \leftarrow ((t + (t >> 4) \& 0xf0f0f0f) \cdot 0x1010101) >> 24$.
6. Return $((i << 5) + t - 1)$.

Modified EEA

Algorithm 3. Modified Extended Euclidean Algorithm for multiplicative inverse in $\mathbf{GF}(2^m)$ field.

Input: $a \in \mathbf{GF}(2^m)$, $a \neq 0$, $n = \lceil \frac{m}{w} \rceil$, where n - number of machine words, which takes polynomial; w - wide of machine words,
usually $w = 32$.

Output: $a^{-1} \bmod f(x)$.

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

Modified EEA

5. $t \leftarrow t \,|\, (t >> 1)$, $t \leftarrow t \,|\, (t >> 2)$, $t \leftarrow t \,|\, (t >> 4)$, $t \leftarrow t \,|\, (t >> 8)$, $t \leftarrow t \,|\, (t >> 16)$.
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. $\deg V = m$.
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 $\deg V = \deg U$.

Modified EEA

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

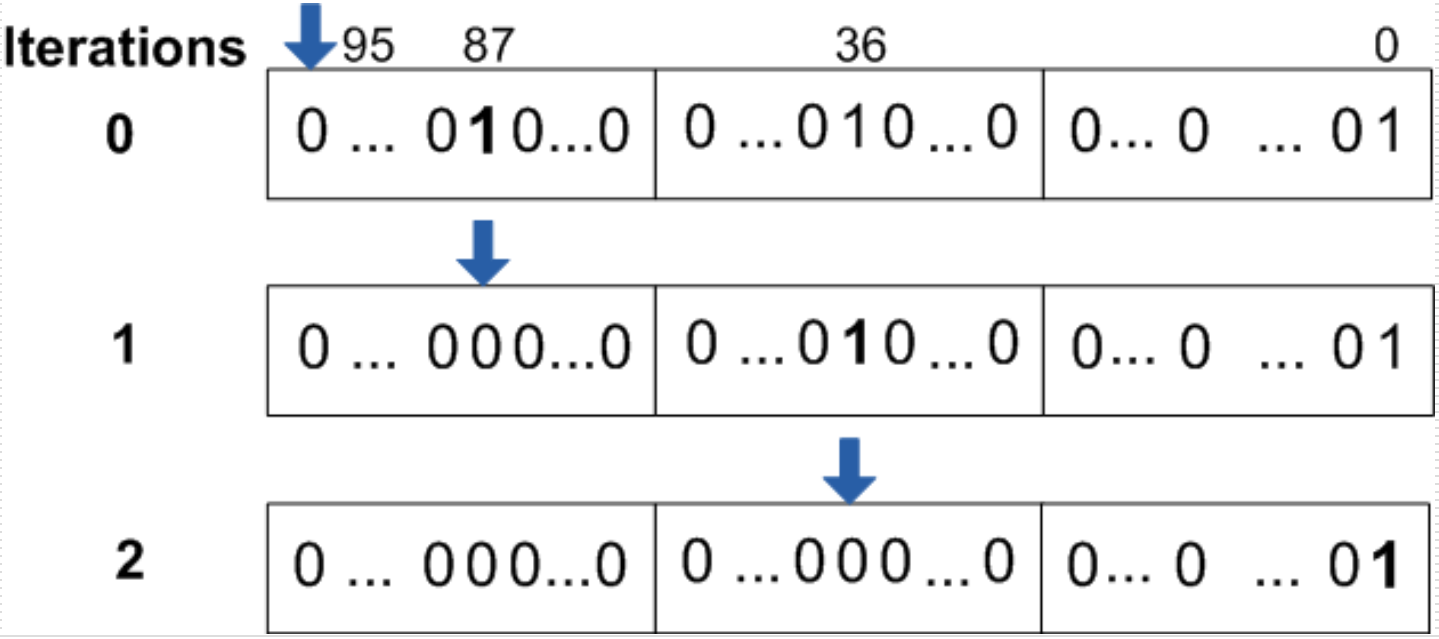
9.9. $t \leftarrow t - ((t >> 1) \,\&\, 0x55555555)$, $t \leftarrow (t \,\&\, 0x33333333) + ((t >> 2) \,\&\, 0x33333333)$, $t \leftarrow (((t + (t >> 4)) \,\&\, 0xf0f0f0f) \cdot 0x1010101) >> 24$.

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

10. Return (b).

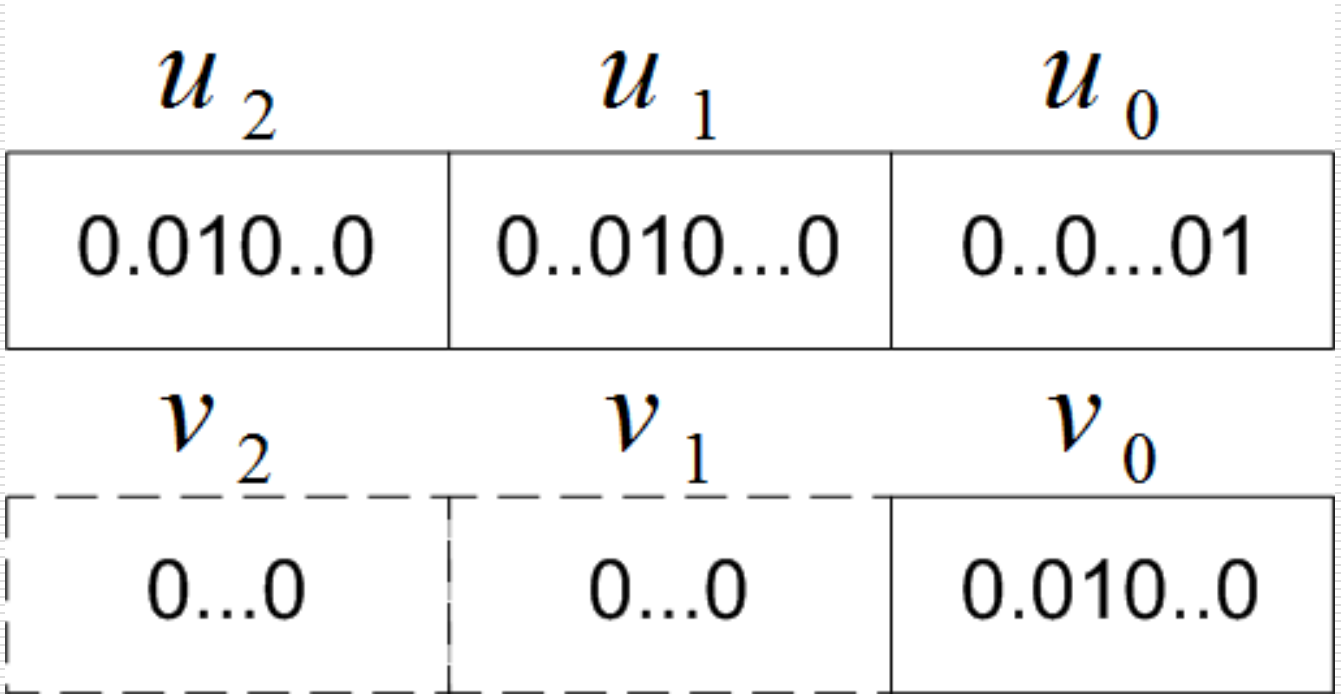
MEEA: Degree of polynomial – next fit index

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



MEEA: Shift of polynomial - significant terms (before)

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



MEEA: Shift of polynomial - significant terms (after)

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

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

MEEA: Addition of polynomials - significant terms

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

	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

MEEA: Complexity

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

$I_{\deg}$ - complexity of algorithm for computing degree of polynomial;

I_{add} - complexity of algorithm for adding two polynomials;

I_{shl} - complexity of algorithm for shifting to an arbitrary number of bits (may to exceed the length of machine word);

I_{swp} - complexity of algorithm for exchange two polynomials.

* k- number of iterations.

MEEA vs EEA

COMPARISON OF COMPUTATIONAL COMPLEXITY

Performance comparison

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

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

MEEA vs EEA

COMPARISON OF PERFORMANCE

Software Implementation

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

Experimental: Terms

- Average time of 1 million operations
- Recommended field:
  - DSTU 4145-2002
  - FIPS 186-3
- Computers:
  - Intel Core i3-350M (notebook)
  - Intel Core i5-3570 (desktop)
  - Intel Core i5-4670 (desktop)

Test binary fields

m	Irreducible polynomial
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	$x^{571}+x^{10}+x^{5}+x^{2}+1$

Performance

m	Time, ms											
	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

* – MEEA.

MEEA vs EEA

CONCLUSIONS

Conclusions

- Deep analysis of EEA allows to identify potential improvements:
  - Allows save degree computing of polynomial ***v(x)***. It is either not changed or equal to degree of ***u(x)***.
  - Allows use algorithm of "next fit index" for computing degree of ***u(x)***, i.e. continual refinement of degree from previous value.
  - Allows use "trick" for computing of MSB in the machine word. Is eliminates the comparisons and branching.

Conclusions

- Identified improvements EEA were implemented in the MAAE.
- Computational complexity MEEA lower at most 2 times than EEA.
- Software implementation of MEEA has higher performance in 15-50% than EEA, with field size growth, in average.

Conclusions

- Proposed implementation of MEEA inversion algorithm has not adjusted for multithreaded execution. It does not allow realize full potential of modern multicore CPU.
- Applying MEEA in software implementation algorithms of digital signature generation and verification according to DSTU 4145-2002, has increased performance:
  - on 16-20% in Affine point representation;
  - on 2-4% in Lopez-Dahab projective representation.

Further researches

- Modern CPU develops in direction of increasing number of commands execution threads. In turn, it requires developing corresponding algorithms for efficient software implementation on perspective CPU.
- NVIDIA company already offers GPU with more than 512 cores, and user-friendly framework CUDA for creation multithreading applications.
- Further direction of researches will sent to analyzing and effective parallelizing of arithmetic algorithms in **GF**(2^m) field.

Questions?

Thank you for your attention!

National Aviation University

Chair of information security systems

Mary Bulakh

email: bulakh.masha@gmail.com