Accelerating Parallelized Pollard Rho to Identify Weak Class Elliptic Curves

Intan Muchtadi, Budi Rahardjo
Marisa Paryasto, Tomy Ardiansyah,
Sa’aadah Sajjana Carita

Faculty of Maths and Natural Sciences
School of Electrical Engineering and Infomatics
Institut Teknologi Bandung (ITB), Indonesia

MOTIVATION

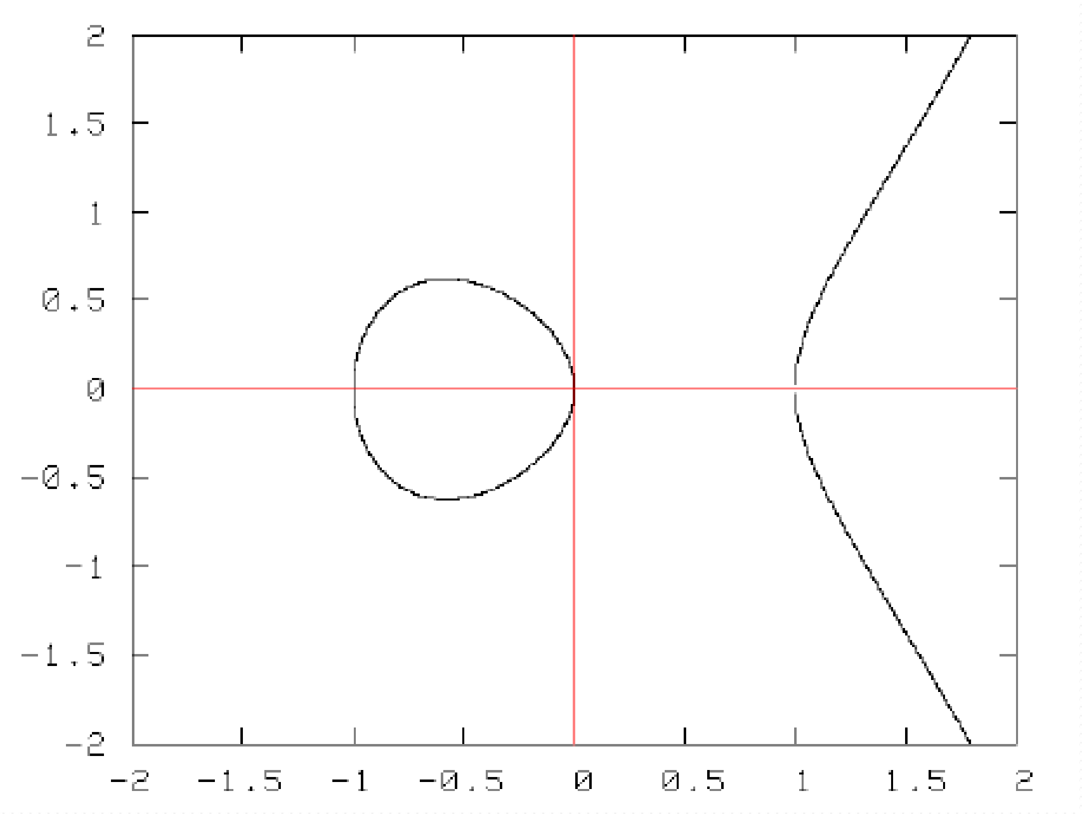
Devices with limited sources are easy to get wired /attacked

We need cryptography implementation on these kinds of devices

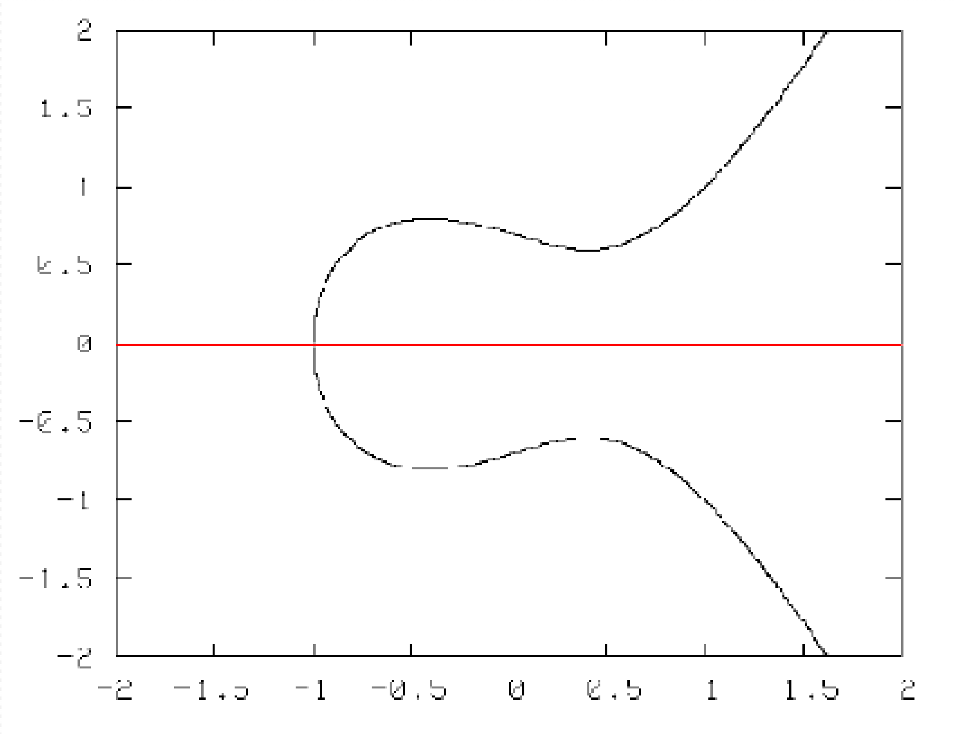
ECC is one of the solution, but ECC needs big computation

Elliptic curve

$$y^2 = x^3 - x$$

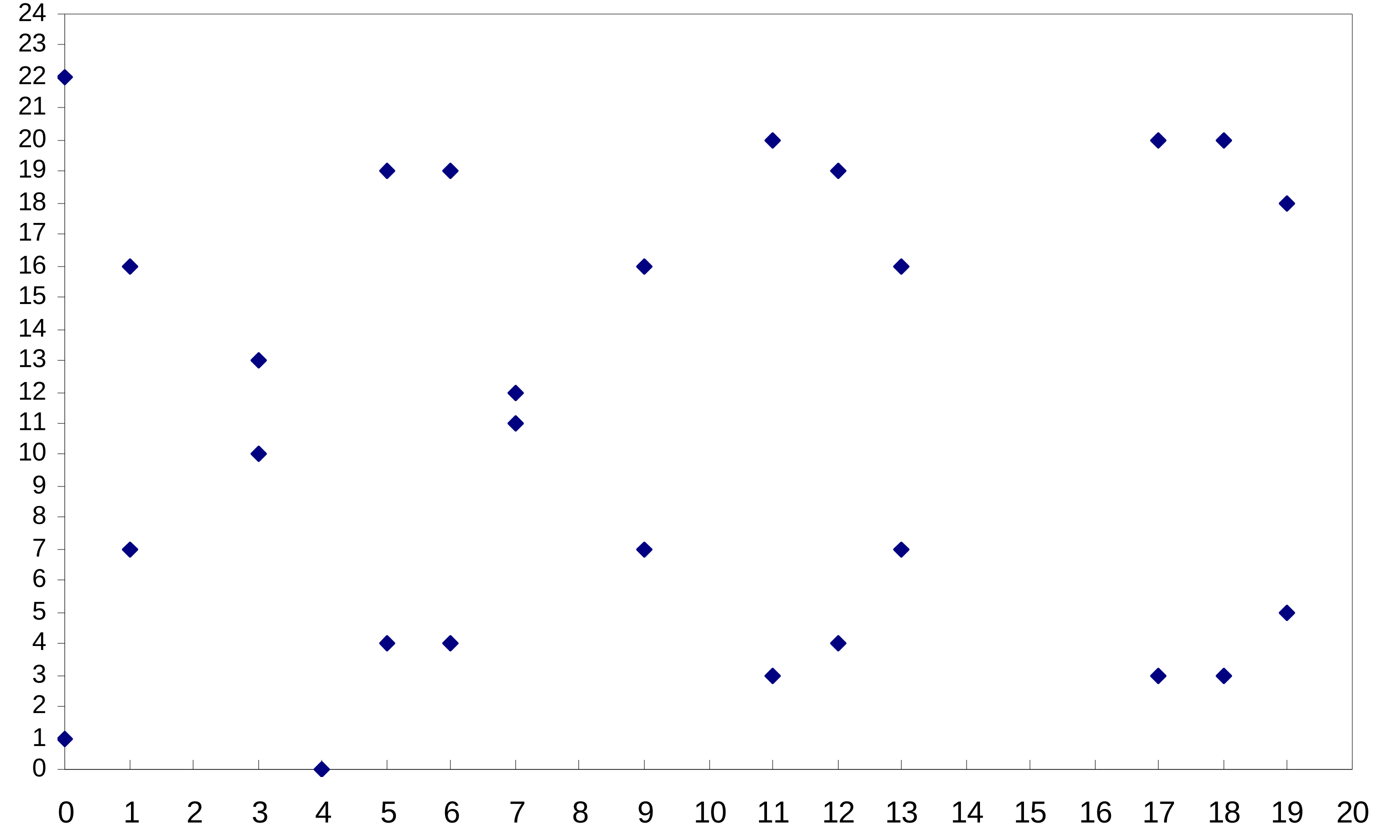


$y^2 = x^3 - ½x + ½$

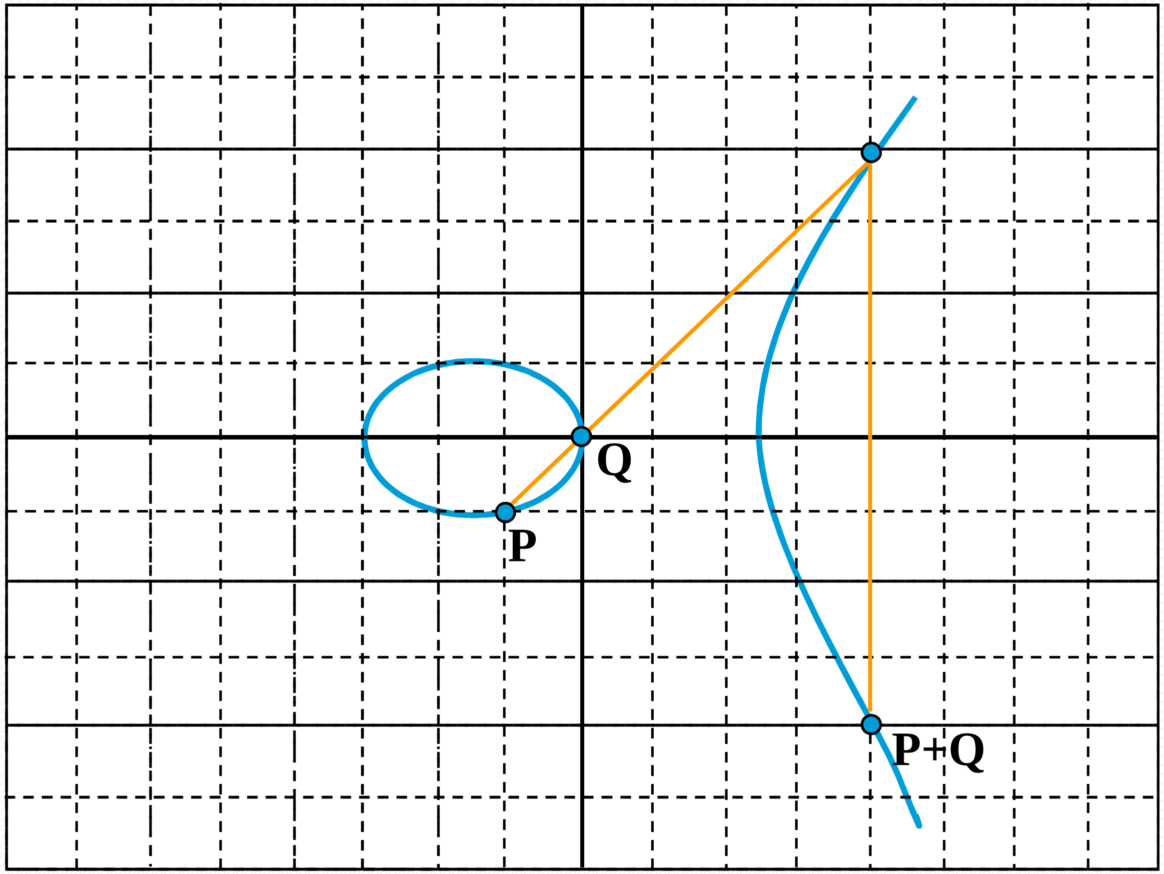


Elliptic curve over F_{23}

$y^2 = x^3 + x + 1$



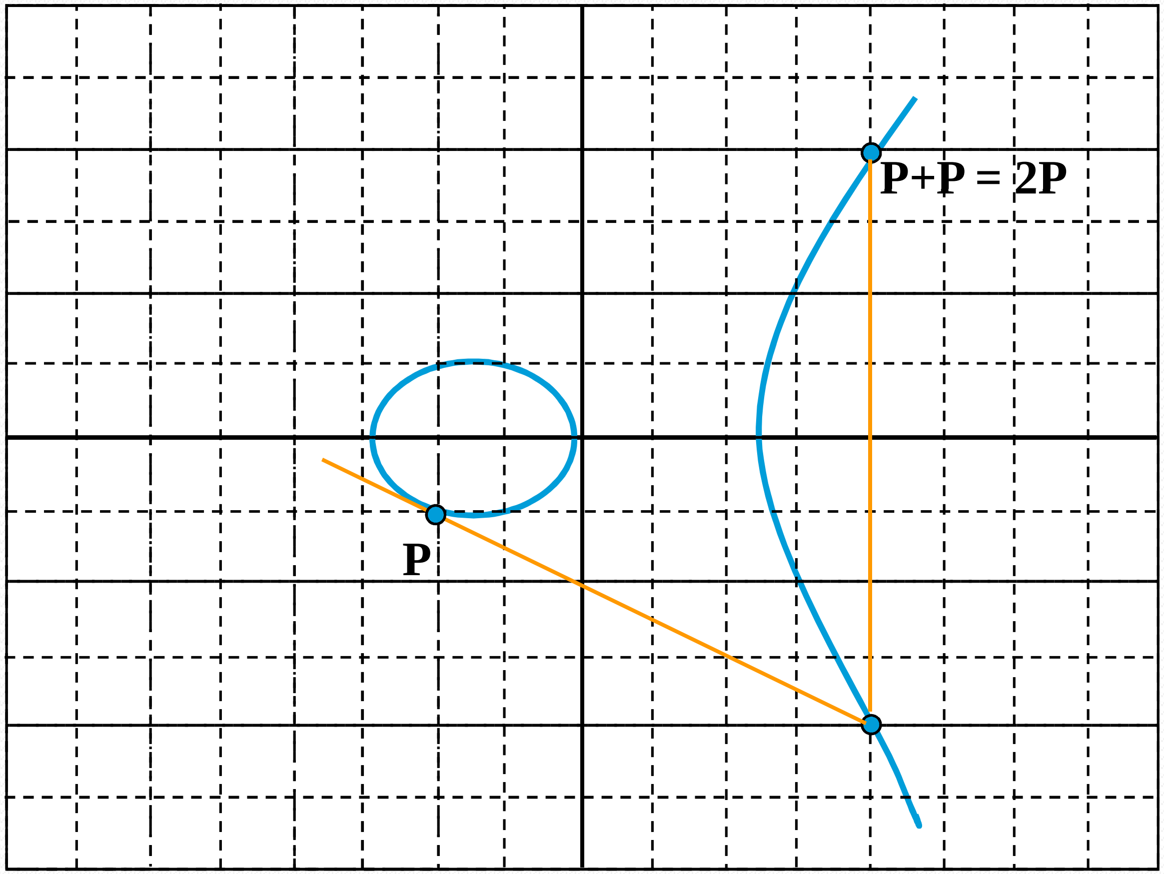
Elliptic Curve Addition



Multiples in Elliptic Curves 1

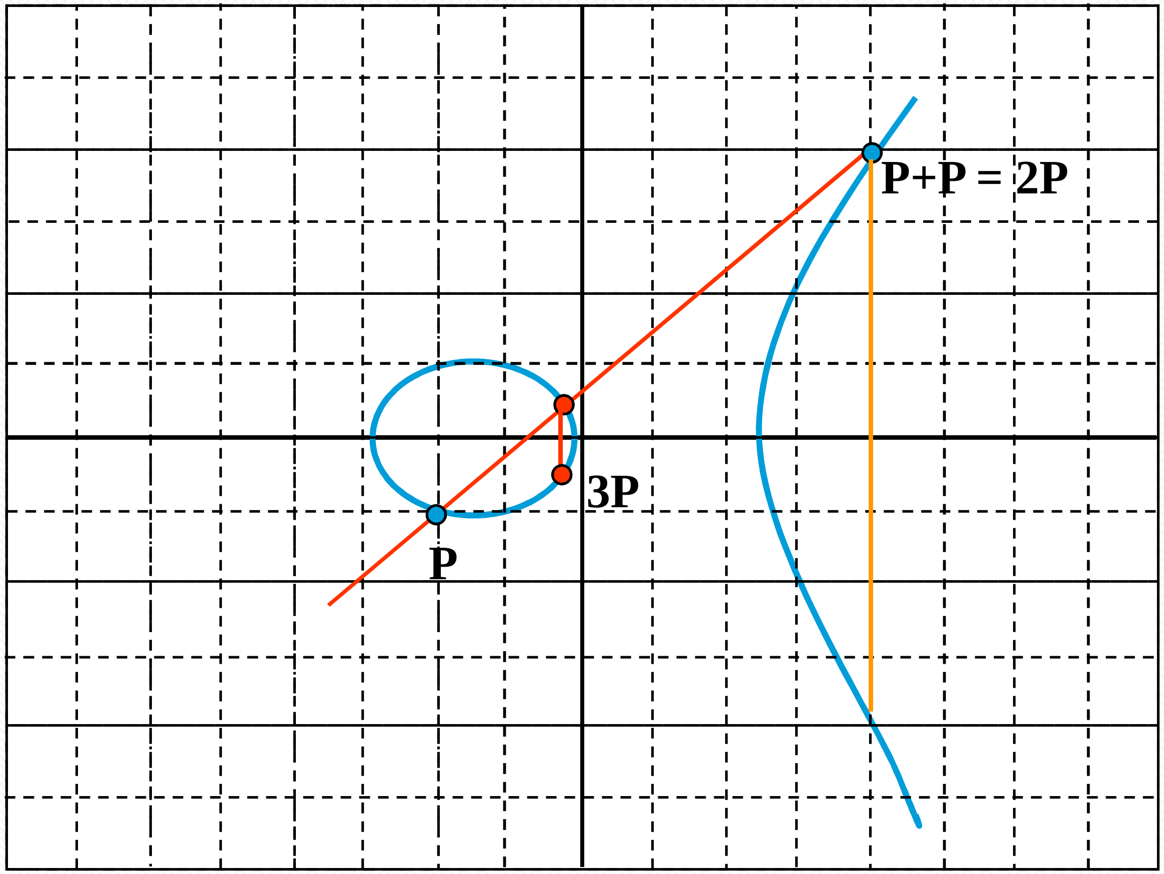
- The interest in Elliptic Curve Addition is the process of adding a point to itself.
  - That is given a point P find the point P+P or 2P.
  - This is done by drawing a line tangent to P and reflecting the point at which it intercepts the curve
  - P can be added to itself k times resulting in a point W = kP.

Multiples in Elliptic Curves 1



Multiples in Elliptic Curves 2

- Finding the value of 3P:



Elliptic Curve Encryption

- INPUT: Prime p, elliptic curve E, point P of order n, private key $d \in [1,n-1]$, plaintext m
- OUTPUT: Cipher text (C_1,C_2)

1. Compute $Q=dP$
2. Represent the message m as the point M in $E(F_p)$
3. Select $k \in [1,n-1]$
4. Compute $C_1 = kP$
5. Compute $C_2 = M + kQ$
6. Return (C_1,C_2)

Elliptic Curve Decryption

- INPUT : prime p, elliptic curve E, point P of order n, private key d, ciphertext (C_1, C_2)
- OUTPUT: Plaintext m

1. Compute $M = C_2 - dC_1$ and extract m from M
2. Return (m).

$(M = C_2 - dC_1 = M + kQ - dkP = M + kdP - dkP)$

Elliptic Curve Security

- The security of the Elliptic Curve algorithm is based on the fact that it is very difficult (as difficult as factoring) to solve the Elliptic Curve Discrete Logarithm Problem:

Given two points P and Q where $Q = kP$, find the value of k

POLLARD RHO

Let $G = E(\mathbb{F})$, with $|P| = M$, and P and Q such that $Q = [k]P$ in G. We aims to find k.

The Algorithm

1. By using a hash function, we divide G into 3 sets, S_1, S_2, S_3 with almost equal number of elements, but $\mathcal{O} \notin S_2$.
2. Define an iteration function f:

$$R_{i+1} = f(R_i) = \begin{cases} P + R_i, R_i \in S_1 \\ 2R_i, \quad R_i \in S_2 \\ Q + R_i, R_i \in S_3 \end{cases} \quad (1)$$

Since $R_{i+1} = 2R_i$ if $R_i \in S_2$, then if $\mathcal{O}$ is in S_2, in some time $R_i = \mathcal{O}$ and the values of the iteration functions will all be $\mathcal{O}$. That is why we makethe assumption of $\mathcal{O} \notin S_2$.

3. Let $R_i = s_iP + t_iQ$, then

$$s_{i+1} = \begin{cases} s_i + 1 & , R_i \in S_1 \\ 2s_i \bmod m & , R_i \in S_2 \\ s_i & , R_i \in S_3 \end{cases} \quad (2)$$

and

$$t_{i+1} = \begin{cases} t_i & , R_i \in S_1 \\ 2t_i \bmod m & , R_i \in S_2 \\ t_i + 1 & , R_i \in S_3 \end{cases} \quad (3)$$

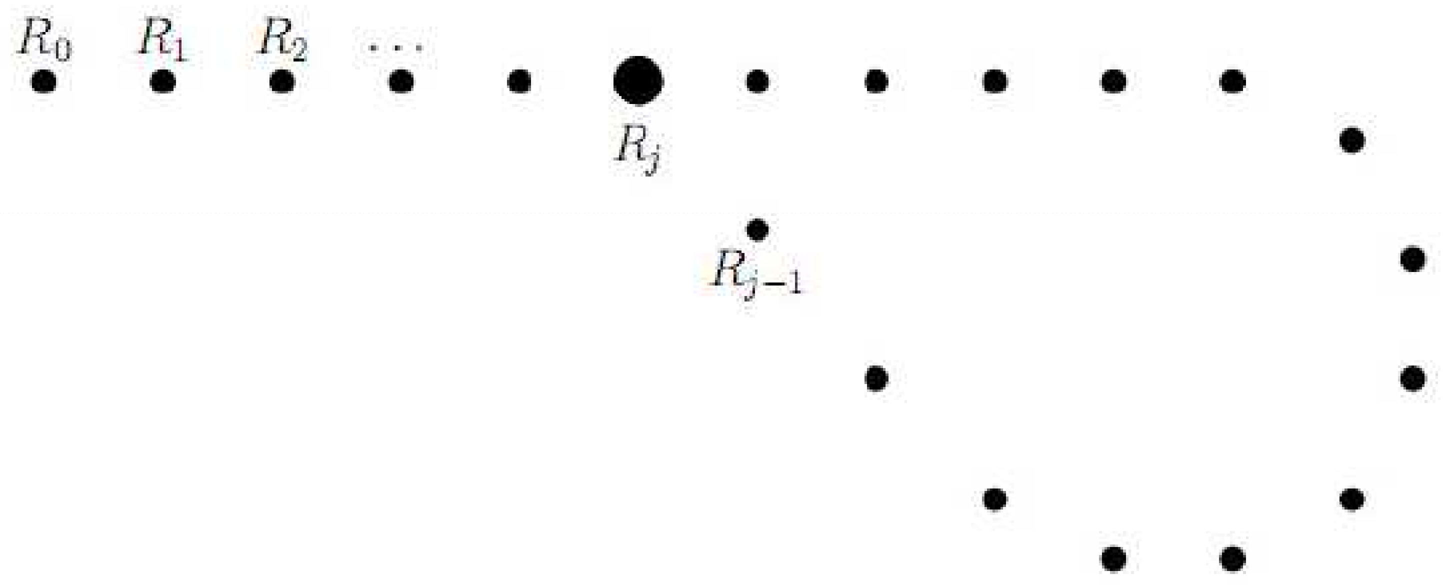
4. Beginning with $R_0 = P, s_0 = 1, t_0 = 0$ we generate R_i until we find $R_j = R_l$ with $j \neq l$.
   When we reach that equality, we will get

$$R_j = s_jP + t_jQ \text{ and } R_l = s_lP + t_lQ \quad (4)$$

   And hence k is

$$k = \frac{s_l - s_j}{t_j - t_l} \bmod m \quad (5)$$

This algorithm can solve the ECDLP in $\mathcal{O}\sqrt{m}$ operation[10].(By the birthday paradox, the expected number of iterations for ordinary Pollard Rho is $\sqrt{\frac{\pi m}{2}}$)[1].


R_0
R_1
R_2
$\cdots$
R_j
R_{j-1}

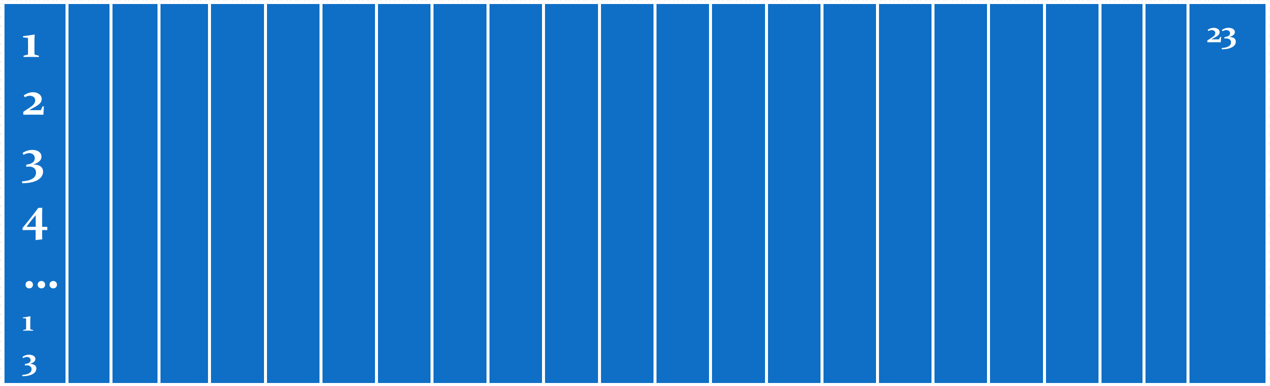
Finite Field

- Operations over the real numbers are slow and inaccurate due to round-off error

- Need to be faster and accurate
- Accurate and efficient :
  - Prime field GF(p)
  - Binary field $GF(2^m)$
  - Composite Field $GF((2^m)^n)$

COMPOSITE FIELD

- Using composite field, we may divide the computation into subfields from GF(2^k) into GF($(2^n)^m$) where k = nm.

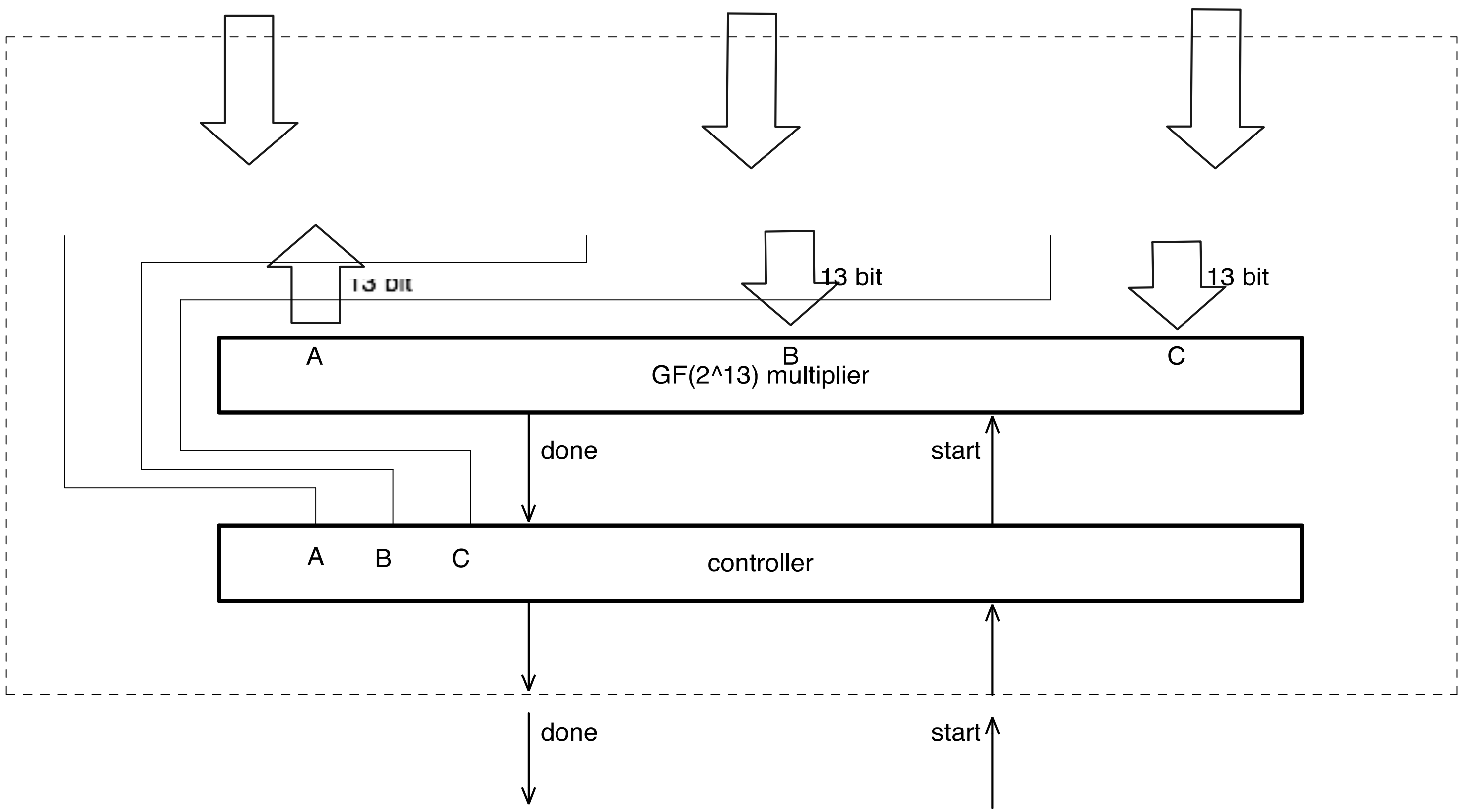


MULTIPLIER

- MULTIPLIER :
  - Create/improve algorithms
  - Design implementation

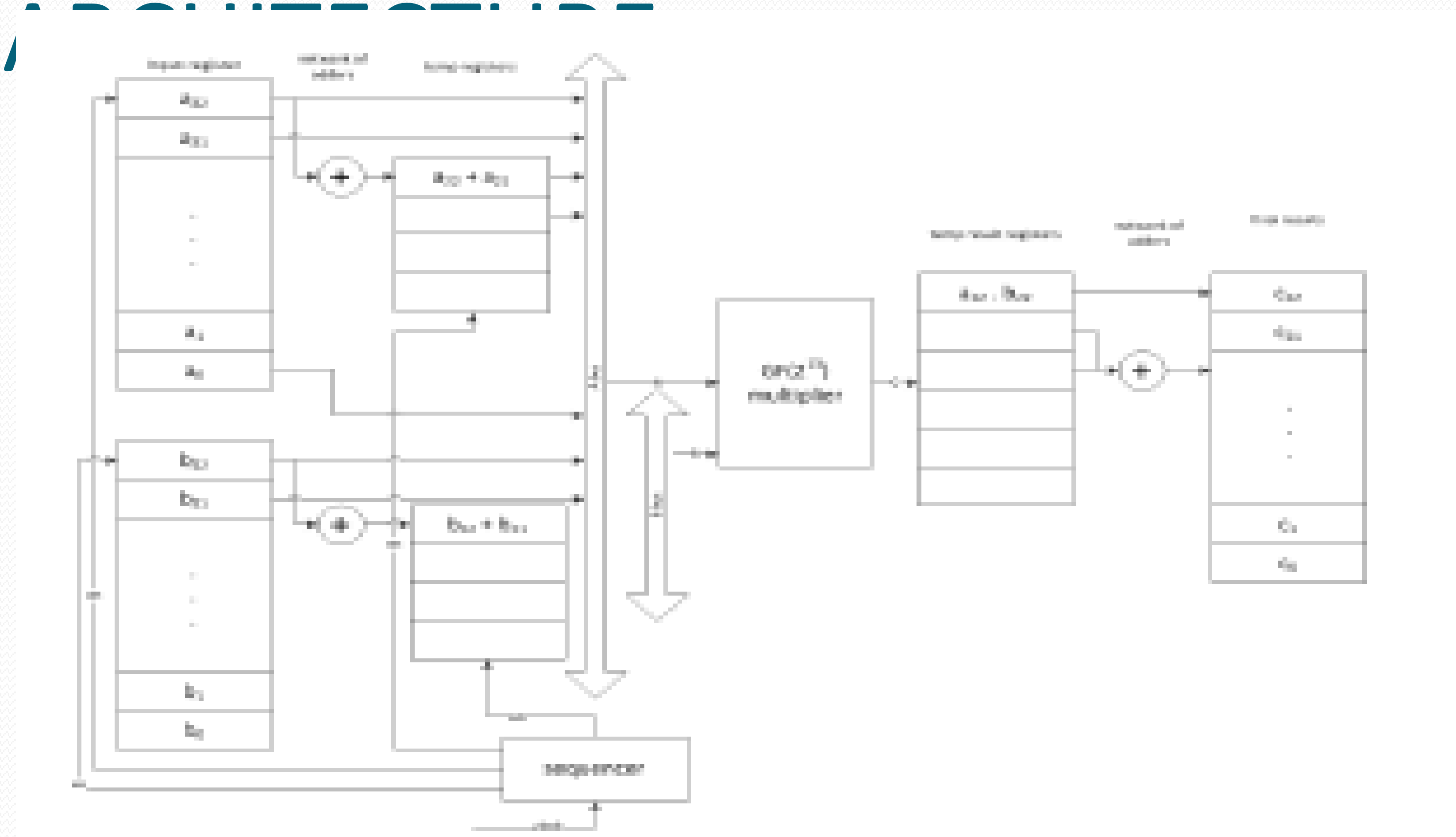
- LUT is used for multiplication in ground field $GF(2^{13})$ and Karatsuba Offman Algorithm for the extension field multiplication $GF(2^{13})^{23}$

Multiplier for GF(2^13)



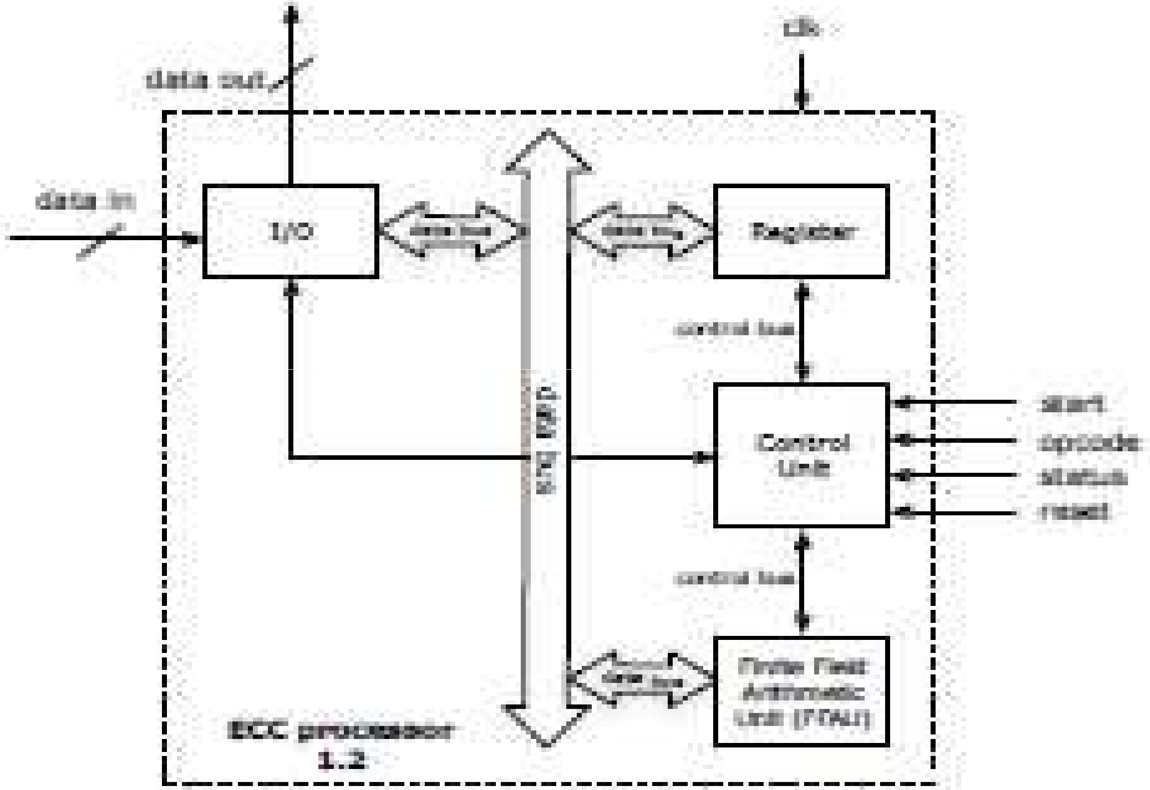
[Paryasto-Rahardjo-Muchtadi-Kuspriyanto2010]

MULTIPLIER GENERAL ARCHITECTURE



[Paryasto-Rahardjo-Yuliawan-Muchtadi-Kuspriyanto2012]

ECC ARCHITECTURE WITH COMPOSITE FIELD



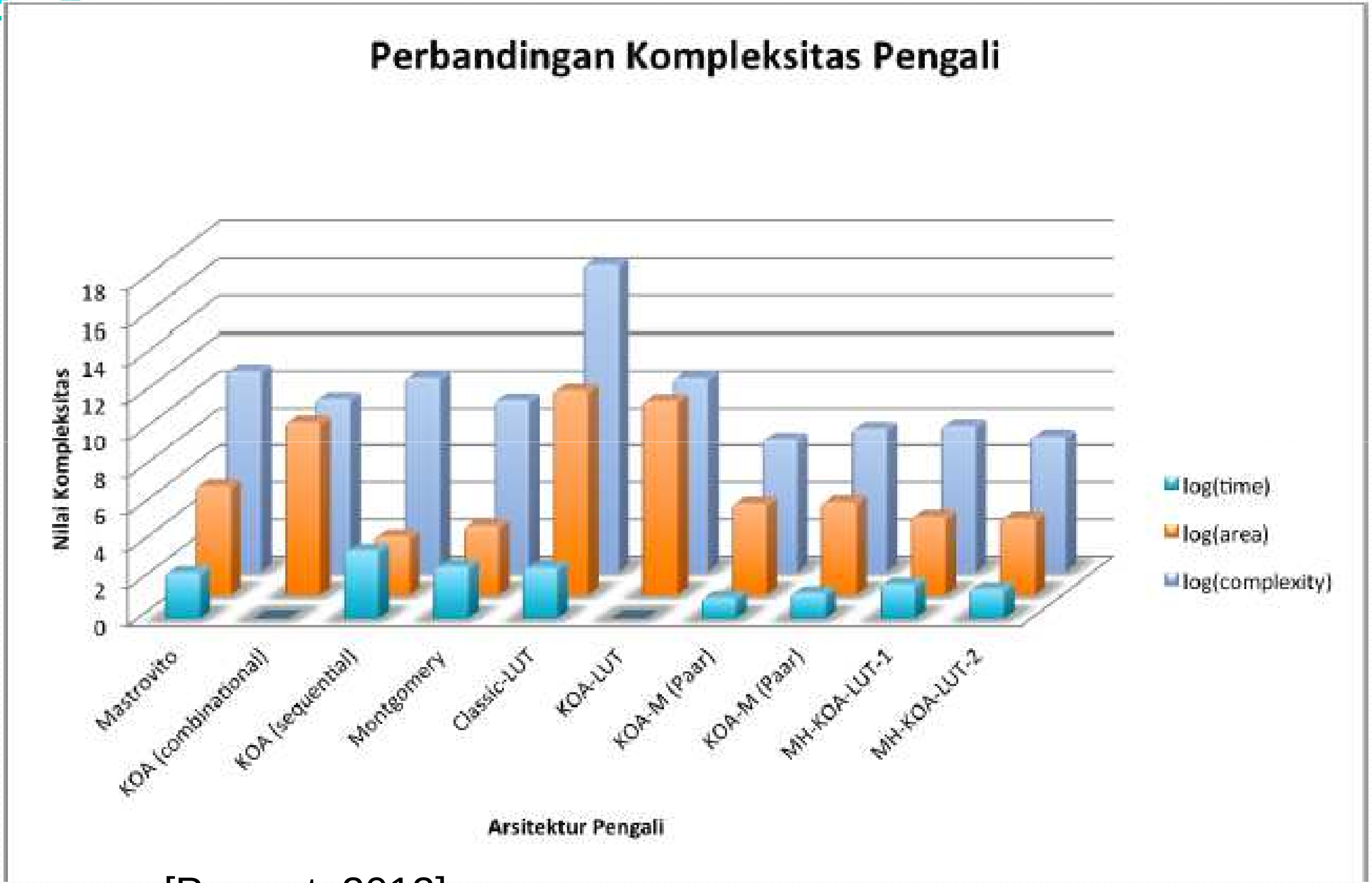
[Paryasto-Rahardjo-Muchtadi-Kuspriyanto2011] i fixed

	Config	n	m	Area	Time	Complexity	log(time)	log(area)	log(complexity)
1	Mastrovito	299		805058	299	71972945558	2.4756712	5.9058269	10.85716928
2	KOA (combinational)	299		2243215876	1	2243215876	0	9.3508711	9.35087107
3	KOA (sequential)	299		1512	4608	32105299968	3.6635125	3.1795518	10.50657673
4	Montgomery	299		5385	598	1925697540	2.7767012	3.7311857	9.284588076
5	Classic-LUT	13	23	1.1538E+11	529	3.22872E+16	2.7234557	11.062119	16.50903036
6	KOA-LUT	13	23	3.1407E+10	1	31407035559	0	10.497027	10.49702695
7	KOA-M (Paar)	13	23	87351	13	14762319	1.1139434	4.9412679	7.169154586
8	KOA-M (Paar)	23	13	105774	23	55954446	1.3617278	5.0243789	7.7478346
9	MH-KOA-LUT-1	13	23	21707	69	103347027	1.8388491	4.3365998	8.014297988
10	MH-KOA-LUT-2	23	13	305172	39	464166612	1.5910646	5.4845447	8.666673898

l fixed

	Config	n	m	Area	Time	Complexity	log(time)	log(area)	log(complexity)
1	Mastrovito	299		805058	299	71972945558	2.4756712	5.9058269	10.85716928
2	KOA (combinational)	299		2243215876	1	2243215876	0	9.3508711	9.35087107
3	KOA (sequential)	299		1512	4608	32105299968	3.6635125	3.1795518	10.50657673
4	Montgomery	299		5385	598	1925697540	2.7767012	3.7311857	9.284588076
5	Classic-LUT	13	23	1.1538E+11	529	3.22872E+16	2.7234557	11.062119	16.50903036
6	KOA-LUT	13	23	3.1407E+10	1	31407035559	0	10.497027	10.49702695
7	KOA-M (Paar)	13	23	87351	13	14762319	1.1139434	4.9412679	7.169154586
8	KOA-M (Paar)	23	13	105774	23	55954446	1.3617278	5.0243789	7.7478346
9	MH-KOA-LUT-1	13	23	25884	69	123233724	1.8388491	4.4130314	8.090729573
10	MH-KOA-LUT-2	23	13	19779	39	30083859	1.5910646	4.2962043	7.478333545

[Paryasto2012]



[Paryasto2012]

Result 1 [Muchtadi2012]

- Speed up the Pollard Rho algorithm for elliptic curves over composite fields, by using the multiplier that combines the LUT and KOA proposed in [Paryasto2012]

Elliptic Curves over GF(2^n)

Elliptic curve over GF(2^n) is defined with Weierstrass equation, which after transformed by admissible change of variables, can be written as

$$E(GF(2^n)) = \{(x, y) \in GF(2^n)^2 : y^2 + xy = x^3 + ax^2 + b\} \cup \{O\},$$

- where *O* is the projective closure of the equation .

Modified Pollard Rho

To speed up Pollard Rho, the iterating function f is defined on the equivalence class rather than just one point in $\langle P \rangle$.

The expected number of iterations for the modified Pollard Rho algorithm is $\sqrt{\frac{\pi m}{2t}}$.

Negation Map

If $P = (x, y)$, we have $-P = (x, x + y)$. The negation map $\psi(P) = -P$ has order 2, thus the number of iterations is expected to be

$$\frac{\sqrt{\pi n}}{2}.$$

This map is applicable to all elliptic curve.

Frobenius Map

The Frobenius map can be used only for Koblitz curves. A Koblitz curve E_a (where $a \in \{0,1\}$) is an elliptic curve defined over $GF(2^m)$:

$$E_a : y^2 + xy = x^3 + ax^2 + 1.$$

The Frobenius map $\tau: E_a(GF(2^m)) \to E_a(GF(2^m))$ is defined by $\tau(O) = O$ *and* $\tau(x, y) = (x^2, y^2)$.

Pollard Rho algorithm using equivalence classes under the Frobenius map has an expected number of iterations

$$\sqrt{\frac{\pi n}{2m}}$$

as the order of the map is n.

Furthermore, for Koblitz curves, the Pollard Rho's algorithm can exploit both the Frobenius and negation map to achieve an expected running time of

$$\frac{1}{2}\sqrt{\frac{\pi n}{m}}$$

Experimental Results1 [Muchtadi-Ardiansyah-Carita2013a]

Let K= $GF(2^7)$.
E=Elliptic Curve defined by $y^2 + x*y = x^3 + x^2 + 1$
Cardinality of E=142
Let $P=R_0=(3,85)$ on E. The order of P is 71.
We choose Q=50P

By using SAGE we compute R_i , as presented below:

i	R_i	$R_i=s_iP+t_iQ$	$-R_i$
0	(3,85)	1P+0Q	(3,86)
1	(19,29)	1P+1Q	(19,15)
2	(99,5)	2P+1Q	(99,102)
3	(65,119)	3P+1Q	(65,54)
4	(5,119)	3P+2Q	(5,114)
5	(49,23)	3P+3Q	(49,38)
6	(75,126)	4P+3Q	(75,53)
7	(55,56)	4P+4Q	(55,15)
8	(65,119)	8P+8Q	(65,54)

We need 8 iterations to get $R_3=R_8$. We get

$$3P + 1Q = 8P + 8Q$$
$$-5P = 7Q$$
$$66P = 7kP$$
$$k = (66/7) \bmod 71 = 50$$

Computation of Equivalence classes

The following is the computation of equivalence class of R_1, R_2, R_3, and R_4.

i		$\psi^i(R_1)$
1	$(R_1)^2$	(3,86)
2	$(R_1)^4$	(5,114)
3	$(R_1)^8$	(17,122)
4	$(R_1)^{16}$	(7,58)
5	$(R_1)^{32}$	(21,90)
6	$(R_1)^{64}$	(23,96)

i		$\psi^i(R_2)$
1	$(R_2)^2$	(125,17)
2	$(R_2)^4$	(47,7)
3	$(R_2)^8$	(77,21)
4	$(R_2)^{16}$	(49,23)
5	$(R_2)^{32}$	(31,19)
6	$(R_2)^{64}$	(83,3)

Equivalence class (contd)

i		$\psi^i(R_5)$
1	$(R_3)^2$	(97,107)
2	$(R_3)^4$	(121,61)
3	$(R_3)^8$	(63,79)
4	$(R_3)^{16}$	(75,53)
5	$(R_3)^{32}$	(37,15)
6	$(R_3)^{64}$	(9,85)

i		$\psi^i(R_7)$
1	$(R_4)^2$	(17,107)
2	$(R_4)^4$	(7,61)
3	$(R_4)^8$	(21,79)
4	$(R_4)^{16}$	(23,53)
5	$(R_4)^{32}$	(19,15)
6	$(R_4)^{64}$	(3,85)

Result with Frobenius

We can see that $\psi^6(R_x) = R_y = P$.

Therefore $\psi^6(3P + 2Q) = P$. Since $\lambda = 103$,

$3\psi^6(P) + 2\psi^6(Q) = P$

$3\lambda^6 P + 2\lambda^6 kP = P$

Since $\lambda = 103$, we get $3(103)^6 + 2(103)^6 k = 1$,

therefore $k = ((1-3(103)^6)/2(103)^6) \bmod 71 = 50$.

Therefore, in this case, by using Frobenius map we just need 4 iterations to find two collision points. Note that we also need 6 times 4 squarings.

Result with Frobenius-Negation

Notice that –P=(3,86), hence $\psi(R_1) = (R_1)^2 = -P$.

$\psi(P + Q) = -P$

$$103P + 103kP = -P$$

$$k = ((-1-103)/103) \bmod 71 = 50.$$

Therefore by Frobenius-Negation map we just need **one** iteration to get collision points.

Comparison

Method	By experiment	By formule
	#of iterations	# of iterations
Ordinary	8	10.55 ~11
Negation	8	7,46 ~ 8
Frobenius	4	3,99 ~ 4
Frob-Neg	1	2,82~3

Experimental Result 3, using Random Walk [Muchtadi-Ardiansyah-Carita2013c]

i	X(i)[0]	X(i)[1]	s_i	t_i
0	3	85	1	0
1	17	107	1	2
2	95	14	1	6
3	99	102	3	6
4	63	112	3	9
5	69	50	3	13
6	99	102	8	13

i	psi^i	X(1)[0]	X(1)[1]
1	[X(1)]^2	7	61
2	[X(1)]^4	21	79
3	[X(1)]^8	23	53
4	[X(1)]^16	19	15
5	[X(1)]^32	3	85
6	[X(1)]^64	5	119

Comparison

Method	By experiment	By formule
Ordinary	8	11
Negation	8	8
Frobenius	4	4
Frob-neg	1	3
Random Walk	6	11
Frob-Random Walk	1	4

Random Walk with new point

Pollard Rho biasa

i	X(i)[0]	X(i)[1]	s_i	t_i
0	5	114	1	0
1	65	119	1	1
2	11	94	1	2
3	67	41	1	3
4	113	107	2	3
5	97	107	2	4
6	127	61	2	5
7	75	126	4	10
8	23	34	4	11
9	127	66	5	11
10	75	53	10	22
11	51	84	20	44
12	21	90	40	17
13	7	61	40	18
14	31	12	9	36
15	37	15	10	36
16	67	41	11	36

Adding Walks

i	R(i)[0]	R(i)[1]	s_i	t_i
0	5	114	1	0
1	51	103	1	4
2	9	85	1	7
3	21	90	1	9
4	19	28	1	11
5	23	34	4	11
6	3	85	7	11
7	37	42	7	13
8	121	68	11	13
9	57	13	22	26
10	113	26	24	26
11	31	19	27	26
12	11	85	29	26
13	5	119	29	28
14	125	17	29	33
15	93	53	31	33
16	19	15	36	33
17	63	112	38	33
18	75	126	38	37
19	49	23	38	42
20	65	54	40	42
21	0	1	45	42
22	5	114	46	42

i	psi^i	X(1)[0]	X(1)[1]
1	[X(1)]^2	97	107
2	[X(1)]^4	121	61
3	[X(1)]^8	63	79
4	[X(1)]^16	75	53
5	[X(1)]^32	37	15
6	[X(1)]^64	9	85

psi^i	X(2)[0]	X(2)[1]
[X(2)]^2	69	50
[X(2)]^4	113	26
[X(2)]^8	127	66
[X(2)]^16	43	100
[X(2)]^32	93	104
[X(2)]^64	55	56

i	psi^i	X(3)[0]	X(3)[1]
1	[X(3)]^2	101	89
2	[X(3)]^4	105	39
3	[X(3)]^8	57	13
4	[X(3)]^16	95	81
5	[X(3)]^32	51	103
6	[X(3)]^64	27	109

i	psi^i	X(4)[0]	X(4)[1]
1	[X(4)]^2	127	61
2	[X(4)]^4	43	79
3	[X(4)]^8	93	53
4	[X(4)]^16	55	15
5	[X(4)]^32	11	85
6	[X(4)]^64	69	119

psi^i	X(5)[0]	X(5)[1]
[X(5)]^2	121	61
[X(5)]^4	63	79
[X(5)]^8	75	53
[X(5)]^16	37	15
[X(5)]^32	9	85
[X(5)]^64	65	119

Comparison

Method	By experiment	By formule
Ordinary	16	11
Negation	9	8
Frobenius Random Walk	5	4
Frob-neg Random Walk	4	3
Random Walk	22	11

Speeding the Squaring using Normal Basis

- A polynomial basis in $GF(2^n)$ is a basis of the form
  $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$

- A normal basis in $GF(2^n)$ is a basis of the form
  $\{\alpha, \alpha^2, \ldots, \alpha^{2^{n-1}}\}$
- In normal basis squaring is just a cyclic shift of the coordinates.

For example

- w = 10110101
- w^2 = 11011010
- w^4 = 01101101

Table 3. Representation of NB in terms of PB

	1	β	β^2	β^3	β^4	β^5	β^6	β^7	β^8	β^9	β^{10}	β^{11}	β^{12}
β	0	1	0	0	0	0	0	0	0	0	0	0	0
β^2	0	0	1	0	0	0	0	0	0	0	0	0	0
β^{2^2}	0	0	0	0	1	0	0	0	0	0	0	0	0
β^{2^3}	0	0	0	0	0	0	0	0	1	0	0	0	0
β^{2^4}	0	0	0	1	1	0	1	1	0	0	0	0	0
β^{2^5}	0	1	1	0	1	1	1	0	1	0	0	0	1
β^{2^6}	0	1	1	0	1	0	0	1	1	0	1	1	0
β^{2^7}	1	0	0	0	0	1	1	0	0	1	0	1	1
β^{2^8}	1	0	0	0	0	1	0	0	1	0	0	1	1
β^{2^9}	0	0	0	1	1	0	0	1	0	1	0	1	0
$\beta^{2^{10}}$	1	0	1	1	0	0	0	0	0	0	1	0	1
$\beta^{2^{11}}$	1	1	0	1	0	0	0	1	1	0	1	0	1
$\beta^{2^{12}}$	1	0	0	0	1	1	1	0	1	0	1	0	1

Let A be the 13×13 matrix whose entries are binary numbers in the above table. If

$x = \begin{pmatrix} B[0] \\ B[1] \\ \vdots \\ B[12] \end{pmatrix}$ is a representation of an element of $GF(2^{13})$ in NB, then $x' = A^T x$ is its representation in PB. Note that A^T is invertible since elements of NB are linearly independent. Therefore, let $x' = \begin{pmatrix} B[0]' \\ B[1]' \\ \vdots \\ B[12]' \end{pmatrix}$ be the representation of an element of $GF(2^{13})$ in PB, and let $(A^T)^{-1}$ be the inverse of A^T, we have $x = (A^T)^{-1} x'$, none other than its representation in PB. Thus, $(A^T)^{-1}$ is the transition matrix from PB to NB.

Experimental Result 2 [Muchtadi-Ardiansyah-Carita2013b]

Let $K = GF(2^{13})$,
E= Koblitz Curve defined by $y^2 + xy = x^3 + x^2 + 1$
Cardinality of $E = 8374$. Let $P = X(0) = (9, 2951)$ on E. The order of P is 4187.

We choose $Q = 100P$

i	X(i)[0]	X(i)[1]	P	Q
0	9	2951	1	0
1	2922	1830	2	0
2	3569	3925	3	0
3	1355	797	6	0
4	2784	7113	7	0
5	1513	7334	7	1
6	1798	2968	7	2
7	2285	902	14	4
8	4858	6501	15	4
9	5798	4122	15	5
10	2107	6345	30	10
11	2179	6815	30	11
12	7980	1542	30	12
13	4119	6523	31	12
14	59	6619	31	13
15	311	1714	31	14
16	7421	818	32	14
17	1101	7584	33	14
18	1269	4920	33	15
19	283	543	66	30
20	6994	6012	67	30
21	592	8158	67	31
22	8084	2721	67	32
23	6024	360	68	32
24	3670	3524	69	32
25	2317	6238	138	64
26	4417	7507	138	65
27	4465	4094	138	66
28	8030	5958	276	132
29	6828	6345	276	133
30	3291	7649	276	134
31	1608	1066	276	135
32	2149	8026	277	135
33	7800	3802	277	136
34	5533	6054	554	272
35	1832	4866	554	273
36	5058	3676	1108	546
37	4509	7711	2216	1092
38	1269	6093	2216	1093
39	1102	6637	2216	1094
40	5587	7884	2216	1095
41	5543	3832	2216	1096
42	6080	2517	245	2192
43	7724	6879	246	2192
44	5832	1606	246	2193
45	2295	7570	247	2193
46	3073	586	247	2194
47	8018	2282	248	2194
48	2576	989	249	2194
49	6511	5297	250	2194
50	1399	2602	500	201
51	2257	3635	501	201
52	3461	3687	1002	402
53	7730	2828	2004	804
54	3884	748	4008	1608

i	X(i)[0]	X(i)[1]	P	Q
55	7657	2837	4009	1608
56	1704	1630	3831	3216
57	5263	6968	3832	3216
58	6744	2654	3832	3217
59	6405	7628	3833	3217
60	3161	3227	3833	3218
61	1750	867	3479	2249
62	3668	6778	3480	2249
63	1421	3051	3480	2250
64	3365	3653	2773	313
65	5263	4023	1359	626
66	4870	7842	2718	1252
67	8130	2175	2718	1253
68	7249	5243	2719	1253
69	928	5864	1251	2506
70	139	7627	1251	2507
71	5008	7590	1251	2508
72	7663	3778	1251	2509
73	1149	2737	2502	831
74	1215	5602	817	1662
75	429	2825	817	1663
76	570	4564	1634	3326
77	2628	131	3268	2465
78	6838	6647	3269	2465
79	1251	1604	3269	2466
80	7429	1136	3270	2466
81	6347	2571	3271	2466
82	3908	4526	3272	2466
83	5345	2060	2357	745
84	670	4826	2358	745
85	7365	8155	529	1490
86	67	3068	529	1491
87	6283	4020	1058	2982
88	1996	7385	2116	1777
89	4702	3237	2116	1778
90	6407	5259	45	3556
91	8180	1984	90	2925
92	4555	358	91	2925
93	2604	1050	92	2925
94	5165	4273	93	2925
95	3070	7663	186	1663
96	766	1866	186	1664
97	520	6623	187	1664
98	2594	5020	187	1665
99	53	3126	374	3330
100	6828	6345	748	2473

We need 100 iterations to get $X(29) = X(100)$. We get

$$\begin{aligned}
276P + 133Q &= 748P + 2473Q \\
-2340Q &= 472P \\
1847kP &= 472P \\
k &= (\frac{472}{1847}) \bmod 4187 \\
k &= 100
\end{aligned}$$

We can see $X(65) = (5263, 4023)$ and its negation is $-X(65) = (5263, 6968)$. We get

$$\begin{aligned}
-(3832P + 3216Q) &= 1359P + 626Q \\
-3842Q &= 13565P \\
345kP &= 1004P \\
k &= (\frac{1004}{345}) \bmod 4187 \\
k &= 100.
\end{aligned}$$

The use of Frobenius, Negation and Normal Basis

The following is the computation of equivalence class under Frobenius map of $X(1), X(2), \cdots, X(20)$. We change it first into normal basis representation then we perform the squaring as cyclic shift.

In the following table we write down the elements of $GF(2^{13})$ as decimal. For example, in polynomial basis $\beta^7 + \beta^6 + \beta^4 + \beta^3 = 0000011011000 = 216$ and by Table 3, $\beta^7 + \beta^6 + \beta^4 + \beta^3 = \beta^{2^4} = 0000000010000$ in normal basis.

Table.5. Computation of equivalence class of X(1)

i	psi^i	(X1)^i[0]	(X1)^i[1]
1	[X(1)]^2	487	2604
2	[X(1)]^4	5371	4395
3	[X(1)]^8	169	7367
4	[X(1)]^16	1143	5090
5	[X(1)]^32	6293	4048
6	[X(1)]^64	3942	2325
7	[X(1)]^128	3127	6098
8	[X(1)]^256	7822	1872
9	[X(1)]^512	195	7992
10	[X(1)]^1024	4147	1337
11	[X(1)]^2048	7519	2073
12	[X(1)]^4096	4684	5978

Table.6. Computation of equivalence class of X(2)

i	psi^i	(X2)^i[0]	(X2)^i[1]
1	[X(2)]^2	3700	2354
2	[X(2)]^4	3563	5063
3	[X(2)]^8	3888	3009
4	[X(2)]^16	7459	1428
5	[X(2)]^32	1820	3198
6	[X(2)]^64	3944	3791
7	[X(2)]^128	3171	2200
8	[X(2)]^256	3998	5997
9	[X(2)]^512	6465	563
10	[X(2)]^1024	7832	1637
11	[X(2)]^2048	471	6897
12	[X(2)]^4096	4603	6166

Table.7. Computation of equivalence class of X(3)

i	psi^i	(X3)^i[0]	(X3)^i[1]
1	[X(3)]^2	7453	745
2	[X(3)]^4	584	5911
3	[X(3)]^8	4896	6007
4	[X(3)]^16	8162	887
5	[X(3)]^32	5195	5805
6	[X(3)]^64	1439	4829
7	[X(3)]^128	3131	2653
8	[X(3)]^256	7902	1066
9	[X(3)]^512	4547	2500
10	[X(3)]^1024	2225	1765
11	[X(3)]^2048	4908	6855
12	[X(3)]^4096	8114	7426

Table.8. Computation of equivalence class of X(4)

i	psi^i	(X4)^i[0]	(X4)^i[1]
1	[X(4)]^2	333	7566
2	[X(4)]^4	4233	891
3	[X(4)]^8	6189	5885
4	[X(4)]^16	2576	989
5	[X(4)]^32	5243	4831
6	[X(4)]^64	159	2649
7	[X(4)]^128	355	1082
8	[X(4)]^256	5341	2244
9	[X(4)]^512	1213	1597
10	[X(4)]^1024	2279	2993
11	[X(4)]^2048	568	4244
12	[X(4)]^4096	1568	6524

Table.9. Computation of equivalence class of X(5)

i	psi^i	(X5)^i[0]	(X5)^i[1]
1	[X(5)]^2	6447	2019
2	[X(5)]^4	2764	6667
3	[X(5)]^8	1309	3428
4	[X(5)]^16	3081	3923
5	[X(5)]^32	7130	2342
6	[X(5)]^64	7307	4823
7	[X(5)]^128	946	2585
8	[X(5)]^256	1674	5178
9	[X(5)]^512	3730	4254
10	[X(5)]^1024	6601	6456
11	[X(5)]^2048	7918	3033
12	[X(5)]^4096	5315	1236

Table.10. Computation of equivalence class of X(6)

i	psi^i	(X6)^i[0]	(X6)^i[1]
1	[X(6)]^2	3628	5333
2	[X(6)]^4	7339	1277
3	[X(6)]^8	1970	6375
4	[X(6)]^16	2826	6754
5	[X(6)]^32	5607	6437
6	[X(6)]^64	289	2696
7	[X(6)]^128	1241	5389
8	[X(6)]^256	7415	5459
9	[X(6)]^512	5858	1031
10	[X(6)]^1024	648	3477
11	[X(6)]^2048	790	6756
12	[X(6)]^4096	684	6449

Table.11. Computation of equivalence class of X(7)

i	psi^i	(X7)^i[0]	(X7)^i[1]
1	[X(7)]^2	636	922
2	[X(7)]^4	5680	714
3	[X(7)]^8	5050	4882
4	[X(7)]^16	7824	6886
5	[X(7)]^32	407	6403
6	[X(7)]^64	507	3740
7	[X(7)]^128	5547	6557
8	[X(7)]^256	4465	4094
9	[X(7)]^512	3459	3393
10	[X(7)]^1024	7024	2882
11	[X(7)]^2048	6393	1447
12	[X(7)]^4096	6966	2427

Table.12. Computation of equivalence class of X(8)

i	psi^i	(X8)^i[0]	(X8)^i[1]
1	[X(8)]^2	3656	6792
2	[X(8)]^4	2235	3415
3	[X(8)]^8	4968	2646
4	[X(8)]^16	4002	1135
5	[X(8)]^32	7185	6613
6	[X(8)]^64	704	8126
7	[X(8)]^128	4950	1307
8	[X(8)]^256	2806	3101
9	[X(8)]^512	89	6858
10	[X(8)]^1024	4417	7507
11	[X(8)]^2048	2179	4636
12	[X(8)]^4096	5672	6762

Table.13. Computation of equivalence class of X(9)

i	psi^i	(X9)^i[0]	(X9)^i[1]
1	[X(9)]^2	4760	6430
2	[X(9)]^4	6732	4045
3	[X(9)]^8	7537	2116
4	[X(9)]^16	5656	1547
5	[X(9)]^32	6138	3749
6	[X(9)]^64	784	7388
7	[X(9)]^128	696	4775
8	[X(9)]^256	1558	7961
9	[X(9)]^512	4084	312
10	[X(9)]^1024	3333	1432
11	[X(9)]^2048	6994	3118
12	[X(9)]^4096	7421	8143

Table.14. Computation of equivalence class of X(10)

i	psi^i	(X10)^i[0]	(X10)^i[1]
1	[X(10)]^2	4958	7734
2	[X(10)]^4	2742	1461
3	[X(10)]^8	4185	2175
4	[X(10)]^16	2331	846
5	[X(10)]^32	6022	5100
6	[X(10)]^64	5696	3972
7	[X(10)]^128	1722	6149
8	[X(10)]^256	2962	3664
9	[X(10)]^512	5265	2555
10	[X(10)]^1024	5357	944
11	[X(10)]^2048	445	1678
12	[X(10)]^4096	1471	3714

Table.15. Computation of equivalence class of X(11)

i	psi^i	(X11)^i[0]	(X11)^i[1]
1	[X(11)]^2	5672	3138
2	[X(11)]^4	4858	2975
3	[X(11)]^8	3656	5312
4	[X(11)]^16	2235	1516
5	[X(11)]^32	4968	6462
6	[X(11)]^64	4002	3021
7	[X(11)]^128	7185	1476
8	[X(11)]^256	704	7550
9	[X(11)]^512	4950	5709
10	[X(11)]^1024	2806	1771
11	[X(11)]^2048	89	6803
12	[X(11)]^4096	4417	3090

Table.16. Computation of equivalence class of X(12)

i	psi^i	(X12)^i[0]	(X12)^i[1]
1	[X(12)]^2	1065	3828
2	[X(12)]^4	2497	3549
3	[X(12)]^8	1780	2596
4	[X(12)]^16	7110	4459
5	[X(12)]^32	7643	3271
6	[X(12)]^64	4714	3000
7	[X(12)]^128	3966	4309
8	[X(12)]^256	3447	2429
9	[X(12)]^512	3670	914
10	[X(12)]^1024	2543	650
11	[X(12)]^2048	672	786
12	[X(12)]^4096	1878	700

Table.17. Computation of equivalence class of X(13)

i	psi^i	(X13)^i[0]	(X13)^i[1]
1	[X(13)]^2	6479	7132
2	[X(13)]^4	7884	7327
3	[X(13)]^8	4295	674
4	[X(13)]^16	2169	1874
5	[X(13)]^32	858	7996
6	[X(13)]^64	4860	1321
7	[X(13)]^128	3676	2329
8	[X(13)]^256	2475	6018
9	[X(13)]^512	4784	5712
10	[X(13)]^1024	7692	1978
11	[X(13)]^2048	241	2890
12	[X(13)]^4096	5431	1511

Table.18. Computation of equivalence class of X(14)

i	psi^i	(X14)^i[0]	(X14)^i[1]
1	[X(14)]^2	1349	8170
2	[X(14)]^4	7497	5131
3	[X(14)]^8	4952	5535
4	[X(14)]^16	2722	5217
5	[X(14)]^32	4425	475
6	[X(14)]^64	2243	4523
7	[X(14)]^128	1576	7409
8	[X(14)]^256	2720	5878
9	[X(14)]^512	4429	920
10	[X(14)]^1024	2259	718
11	[X(14)]^2048	1832	4866
12	[X(14)]^4096	2680	7142

Table.19. Computation of equivalence class of X(15)

i	psi^i	(X15)^i[0]	(X15)^i[1]
1	[X(15)]^2	1485	3026
2	[X(15)]^4	7487	1169
3	[X(15)]^8	1612	3255
4	[X(15)]^16	7856	7864
5	[X(15)]^32	1431	1495
6	[X(15)]^64	3195	7291
7	[X(15)]^128	3806	5764
8	[X(15)]^256	2457	5788
9	[X(15)]^512	6068	6108
10	[X(15)]^1024	4932	1796
11	[X(15)]^2048	3058	3642
12	[X(15)]^4096	145	7355

Table.20. Computation of equivalence class of X(16)

i	psi^i	(X16)^i[0]	(X16)^i[1]
1	[X(16)]^2	5798	1724
2	[X(16)]^4	4760	2950
3	[X(16)]^8	6732	5505
4	[X(16)]^16	7537	5429
5	[X(16)]^32	5656	4115
6	[X(16)]^64	6138	6495
7	[X(16)]^128	784	8140
8	[X(16)]^256	696	4127
9	[X(16)]^512	1558	6415
10	[X(16)]^1024	4084	3788
11	[X(16)]^2048	3333	2205
12	[X(16)]^4096	6994	6012

Table.21. Computation of equivalence class of X(17)

i	psi^i	(X17)^i[0]	(X17)^i[1]
1	[X(17)]^2	7633	1839
2	[X(17)]^4	4654	2669
3	[X(17)]^8	8046	298
4	[X(17)]^16	5165	1180
5	[X(17)]^32	4491	3302
6	[X(17)]^64	6385	4025
7	[X(17)]^128	7030	7508
8	[X(17)]^256	6381	4617
9	[X(17)]^512	6694	7035
10	[X(17)]^1024	2357	6332
11	[X(17)]^2048	5074	2855
12	[X(17)]^4096	2768	4534

Table.22. Computation of equivalence class of X(18)

i	psi^i	(X18)^i[0]	(X18)^i[1]
1	[X(18)]^2	6311	7842
2	[X(18)]^4	2658	1171
3	[X(18)]^8	383	3251
4	[X(18)]^16	5517	7848
5	[X(18)]^32	5477	1239
6	[X(18)]^64	275	7331
7	[X(18)]^128	477	2034
8	[X(18)]^256	4543	6922
9	[X(18)]^512	7649	3517
10	[X(18)]^1024	5934	7716
11	[X(18)]^2048	4662	1201
12	[X(18)]^4096	7726	2231

Table.23. Computation of equivalence class of X(19)

i	psi^i	(X19)^i[0]	(X19)^i[1]
1	[X(19)]^2	413	565
2	[X(19)]^4	447	1649
3	[X(19)]^8	1467	7137
4	[X(19)]^16	2091	6606
5	[X(19)]^32	4702	7931
6	[X(19)]^64	2670	5586
7	[X(19)]^128	303	1072
8	[X(19)]^256	1165	2176
9	[X(19)]^512	3559	5677
10	[X(19)]^1024	3936	4843
11	[X(19)]^2048	3107	3913
12	[X(19)]^4096	8094	2146

Table.24. Computation of equivalence class of X(20)

i	psi^i	(X20)^i[0]	(X20)^i[1]
1	[X(20)]^2	7421	818
2	[X(20)]^4	5798	1724
3	[X(20)]^8	4760	2950
4	[X(20)]^16	6732	5505
5	[X(20)]^32	7537	5429
6	[X(20)]^64	5656	4115
7	[X(20)]^128	6138	6495
8	[X(20)]^256	784	8140
9	[X(20)]^512	696	4127
10	[X(20)]^1024	1558	6415
11	[X(20)]^2048	4084	3788
12	[X(20)]^4096	3333	2205

We can see that $\psi(X(16)) = \psi^2(X(20))$.
Therefore $\psi(32P + 14Q) = \psi^2(67P + 30Q)$. It easy to see that

$$32\psi(P) + 14\psi(kP) = 67\psi^2(P) + 30\psi^2(kP)$$
$$32\lambda(P) + 14\lambda(kP) = 67\lambda^2(P) + 30\lambda^2(kP)$$

Now we try to find λ. By [1, Note 3.72] we know that

$$\lambda^2 - \lambda + 2 \equiv 0 \bmod 4187$$
$$\lambda^2 - \lambda - (4185 + (4187.x)) = 0,\ x \in \mathbb{Z}$$
$$\text{Choose } x = 2,\ \lambda^2 - \lambda - 8372 = 0.$$

We can see that $\lambda = 92$ and $\lambda = -91 \equiv 4096 \pmod{4187}$ fulfill above equation. Therefore we may choose $\lambda = 4096$.

Since $\lambda = 4096$, we get $32(4096) + 14(4096)k = 67(4096)^2 + 30(4096)^2k$, hence $k = \frac{868}{1516}$ (mod 4187)=100.

Therefore, in this case, by using Frobenius map we just need 20 iterations to find two collision points. Note that we also need 12 times squaring per iteration, which is easily done by cyclic shift.

Notice that $-\psi^{12}(X(8)) = \psi(X(11)) = (5672, 3138)$.
It means

$$
\begin{aligned}
-\psi^{12}(15P + 4Q) &= \psi(30P + 11Q) \\
-15\lambda^{12}P - 4\lambda^{12}kP &= 30\lambda P + 11\lambda kP \\
(11\lambda + 4\lambda^{12})kP &= -(15\lambda^{12} + 30\lambda)P \\
(11(4096) + 4(4096)^{12})kP &= -(15(4096)^{12} + 30(4096))P \\
k &= \frac{2040}{4146} (\text{mod } 4187) \\
k &= 100
\end{aligned}
$$

Therefore by Frobenius-Negation map we just need 11 iterations to get two collision points.

Comparison

The following is the comparison between the experimental results and the expected number of iterations explained in previous section:

Table 25. Comparison of number of iterations

Method	By Experiment	By Formule
	number of iterations	number of iterations
Ordinary	100	81.09~81
Negation	65	57.34~57
Frobenius	20	22.49~22
Frob-Neg	11	15.9~16

Implementation- for longer bit [Paryasto-Rahardjo2013]

- Algorithm of squaring operation in polynomial basis implemented using C programming language

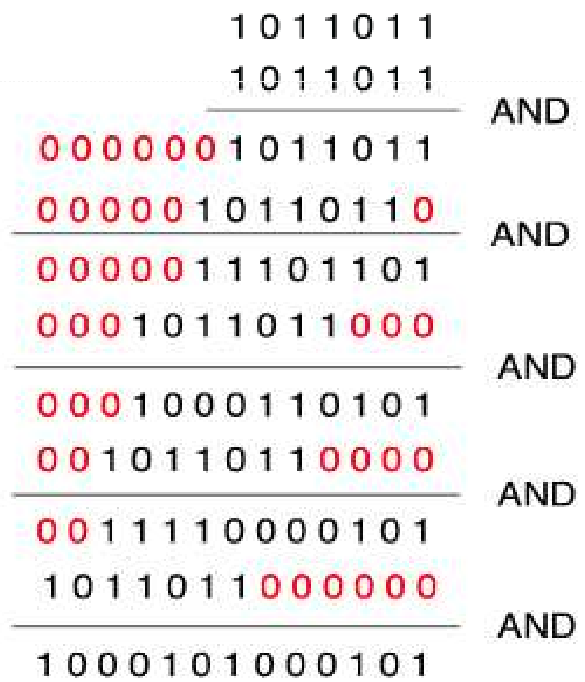


Figure 1. Squaring process

```
void squarePB(int A[], int p[]){
  int i, j, k, x, y;
  int B[m], result[m];
  int p_long[m]; //to hold poly_irred in a longer
      array, making computation easier
  int iteration;
  int flag = 1;

  for (i = 0; i < m; i++)
    p_long[i] = 0;

  for (i = 0; i <= n; i++)
    p_long[i] = p[i];
```

```
    //initializing B[]
    for (i = 0; i < m; i++){
      B[i] = 0;
       result[i] = 0;
    }

    j = n-1;
    for (i = m-1; i >= n-1; i--){
      B[i] = A[j];
       j--;
    }

    if (A[n-1] == 1){
    for (i = 0; i < m; i++)
       result[i] = B[i];

    for (k = n-2; k >= 0; k--){ //if bit 1, shift-lef
          and xor            if (A[k] == 1){
        //shifting B to the left one bit
        for (i = 0; i < m-1; i++)
          B[i] = B[i+1];
          //and make sure to give trailing 0s
                                 B[m-1] = 0;
        //xor
        for (i = 0; i < m; i++)
          result[i] = result[i] ^ B[i];
    }
    else{ //if the bit is 0, just shift no xor
                       //shifting B to the left one bit
    for (i = 0; i < m-1; i++)
      B[i] = B[i+1];

    //and make sure to give trailing 0s
      B[m-1] = 0;
       }
}
```

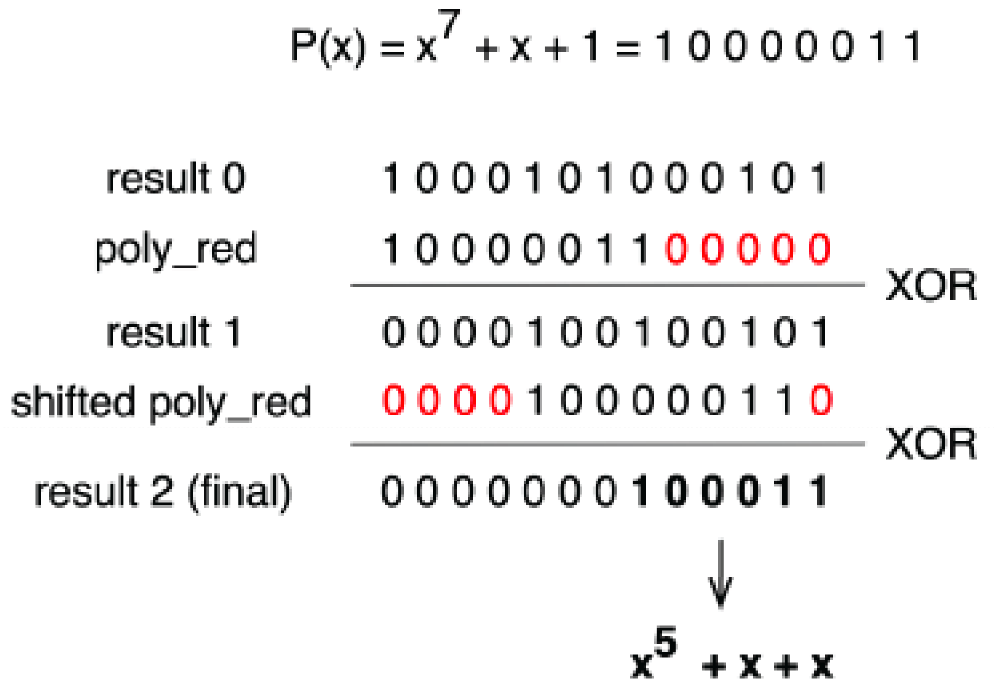


Figure 2. Reduction process

```
//now is the part of reduction
iteration = 0;
k = 0;
while (flag == 1){
  if (k >= m/2)
    flag = 0;

  if (result[k] == 1){ // do xor
    iteration++;

    j = 0;
    for (i = 0; i < m; i++){
      result[i] ^= p_long[j];
      j++;
    }

    k = 0;
  }
  else{ //do shift to the right, no need xor-ing
                       iteration++;

    //shifting result to the right one bit
    for (i = m-1; i >=0; i--)
      p_long[i] = p_long[i-1];

    //make sure to give trailing 0s
    p_long[0] = 0;

        k++;
  }
}
```

Conclusion

- By using Negation and Frobenius map simultaneously we can find two collision points faster than ordinary Pollard Rho.
- Random Walk is not always speeding up the Algorithm, should be combined with Frobenius-Negation.
- Unfortunately Frobenius only works for Koblitz curves
- Koblitz curves could be considered “weak”.
- To speed up the squaring for Frobenius, we suggest the use of normal basis.

OUTPUT (1 proc intl conf, 3 jurnal intl, 1 draft jurnal nas)

- [Muchtadi2012] I.Muchtadi-Alamsyah, Pollard Rho Algorithm for Elliptic Curves over Composite Fields, *Proceeding International Conference on Mathematics and Statistics 2012*, PM 10.

- [Muchtadi-Ardiansyah-Carita2013a] I.Muchtadi-Alamsyah, T.Ardiansyah, S.S.Carita, Pollard Rho Algorithm for Elliptic Curves over GF(2^n) with Negation and Frobenius Map, accepted in *Adv Sciences Letters* **Vol 20** Issue 1, 2014.

- [Muchtadi-Ardiansyah-Carita2013b]] I.Muchtadi-Alamsyah, T.Ardiansyah, S.S.Carita, Pollard Rho Algorithm for Elliptic Curves over GF(2^n) with Negation Map, Frobenius Map and Normal Basis, submitted to Far East Journal of Mathematical Sciences.

OUTPUT

- [Muchtadi-Ardiansyah-Carita2013b]] I.Muchtadi-Alamsyah, T.Ardiansyah, S.S.Carita, Pollard Rho Algorithm for Elliptic Curves over GF(2^n) with Random Walk, Frobenius Map and Normal Basis, submitted to Journal of Software.

- [Paryasto-Rahardjo2013] M.Paryasto, B. Rahardjo, Implementation of Polynomial Basis Squaring, draft.

OUTPUT (Tugas Akhir)

- M. Saputra, Algoritma Pollard Rho dan Modifikasinya pada Kriptografi Kurva Eliptik, Tugas Akhir S1 Matematika ITB, 2012.
- T. Ardiansyah, Algoritma Pollard Rho pada Kurva Eliptik atas Lapangan GF(2^n) dengan Pemetaan Frobenius dan Negasi, Tugas Akhir S1 Matematika ITB, 2013.
- S.S.Carita, Algoritma Pollard Rho pada Kurva Eliptik atas Lapangan GF(2^n) dengan Pemetaan Frobenius, Negasi dan Basis Normal, Tugas Akhir S1 Matematika ITB, 2013.

Presentation

- ICT Asia Regional Meeting, STIC Asie, Bangkok 29-31 October 2012, paper title : Basis Conversion in Composite Field
- International Conference on Mathematics, Statistics and Its Applications, Bali, 19-21 November 2012, paper title: Pollard Rho Algorithm for Elliptic Curves over Composite Fields
- International Conference on Internet Services Technology and Information Engineering, Bogor, 11-12 May 2013, paper title : Pollard Rho Algorithm for Elliptic Curves over $GF(2^n)$ with Negation and Frobenius Map.

ACKNOWLEDGEMENT

- This research is funded by Asahi Glass Foundation Grant based on Surat Perjanjian Pelaksanaan Penelitian Nomor : 2004a/I1.C01/PL/2012.

REFERENCES

- [Muchtadi2012] I.Muchtadi-Alamsyah, Pollard Rho Algorithm for Elliptic Curves over Composite Fields, *Proceeding International Conference on Mathematics and Statistics 2012*, PM 10.

- [Muchtadi-Ardiansyah-Carita2013a] I.Muchtadi-Alamsyah, T.Ardiansyah, S.S.Carita, Pollard Rho Algorithm for Elliptic Curves over GF(2^n) with Negation and Frobenius Map, accepted in *Adv Sciences Letters* **Vol 20** Issue 1, 2014.

- [Muchtadi-Ardiansyah-Carita2013b]] I.Muchtadi-Alamsyah, T.Ardiansyah, S.S.Carita, Pollard Rho Algorithm for Elliptic Curves over GF(2^n) with Negation Map, Frobenius Map and Normal Basis, submitted to Far East Journal of Mathematical Sciences.

References

- [Muchtadi-Ardiansyah-Carita2013b]] I.Muchtadi-Alamsyah, T.Ardiansyah, S.S.Carita, Pollard Rho Algorithm for Elliptic Curves over GF(2^n) with Random Walk, Frobenius Map and Normal Basis, submitted to Journal of Software.

- [Paryasto2012] M.W. Paryasto, Arsitektur Unit Pengali Composite Field Kombinasi MH-KOA untuk Elliptic Curve Cryptography, Disertasi Doktor ITB, 2012.

- [Paryasto-Rahardjo2013] M.Paryasto, B. Rahardjo, Implementation of Polynomial Basis Squaring, draft.

- [Paryasto-Rahardjo-Muchtadi-Kuspriyanto2010] M. W.Paryasto, B.Rahardjo, I. Muchtadi-Alamsyah, Kuspriyanto, *Rancangan Unit Aritmetika Finite Field Berbasis Composite Field*, Prosiding MUNAS Aptikom 2010, pp. 98-102

REFERENCES

- [Paryasto-Rahardjo-Muchtadi-Kuspriyanto2011] M. W.Paryasto, B.Rahardjo, I. Muchtadi-Alamsyah and Kuspriyanto, Implementasi Composite Field pada Elliptic Curve Cryptography, Jurnal Ilmiah Ilmu Komputer Vol.7 No.2 Maret 2011

- [Paryasto-Rahardjo-Yuliawan-Muchtadi-Kuspriyanto2012] M. W. Paryasto, B.Rahardjo, F. Yuliawan, I.Muchtadi-Alamsyah and Kuspriyanto, *Composite Field Multiplier Based on Look-up Table for Elliptic Curve Cryptography Implementation,* ITB Journal of Information and Communication Technology Vol 6 no 1 (2012) 63-81

- THANK YOU VERY MUCH FOR YOUR ATTENTION