부호 기반 양자 내성 암호
ROLLO multiplication 구현 및 양자 프로젝트

2020.3.15

최승주

https://youtu.be/YN1C7oDCoVc

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

ROLLO

3.3 Parameters

The submission of ROLLO-I allows three different levels of security achieving 128, 192 and 256 bits of security according respectively to NIST's security strength categories 1, 3 and 5 [21], we recall them in Table 3. As described in Section 3, the parameters n and m correspond respectively to the degrees of irreducible polynomials P and P_m used to construct the field $\mathbb{F}_{q^m}^n$ and the parameters d and r correspond respectively to the private key and the error's rank.

Param. / Algo.	n	m	d	r	P	P_m	Security level (bits)
ROLLO-I-128	47	79	6	5	$X^{47}+X^5+1$	$X^{79}+X^9+1$	128
ROLLO-I-192	53	89	7	6	$X^{53}+X^6+X^2+X+1$	$X^{89}+X^{38}+1$	192
ROLLO-I-256	67	113	8	7	$X^{67}+X^5+X^2+X+1$	$X^{113}+X^9+1$	256

Table 3. ROLLO-I parameters for each security level

ROLLO / RQC

Algo. \ Param.	n	m	d	r	P	P_m	Security level (bits)
ROLLO-I-128	47	79	6	5	$X^{47}+X^5+1$	$X^{79}+X^9+1$	128
ROLLO-I-192	53	89	7	6	$X^{53}+X^6+X^2+X+1$	$X^{89}+X^{38}+1$	192
ROLLO-I-256	67	113	8	7	$X^{67}+X^5+X^2+X+1$	$X^{113}+X^9+1$	256

Table 3. ROLLO-I parameters for each security level

Instance	P	Π
RQC-I	$X^{67}+X^5+X^2+X+1$	$X^{97}+X^6+1$
RQC-II	$X^{101}+X^7+X^6+X+1$	$X^{107}+X^9+X^7+X^4+1$
RQC-III	$X^{131}+X^8+X^3+X^2+1$	$X^{137}+X^{21}+1$

Table 2: Polynomials considered for RQC. P is the polynomial used to define $\mathbb{F}_{q^m}^n$ as $\mathbb{F}_{q^m}[X]/\langle P\rangle$ and Π is the polynomial used to define $\mathbb{F}_{q^m}$ as $\mathbb{F}_q[X]/\langle\Pi\rangle$.

Classic McEliece / NTS-KEM

```python
from projectq import MainEngine
from projectq.ops import H, CNOT, Swap, Measure, Toffoli
from projectq.backends import CircuitDrawer, ResourceCounter


#

def mul_con(eng):
        a0  = eng.allocate_qubit()
        a1  = eng.allocate_qubit()
        a2  = eng.allocate_qubit()
        a3  = eng.allocate_qubit()

        a4  = eng.allocate_qubit()
        a5  = eng.allocate_qubit()
        a6  = eng.allocate_qubit()
        a7  = eng.allocate_qubit()

        a8  = eng.allocate_qubit()
        a9  = eng.allocate_qubit()
        a10 = eng.allocate_qubit()
        a11 = eng.allocate_qubit()

        b0  = eng.allocate_qubit()
        b1  = eng.allocate_qubit()
        b2  = eng.allocate_qubit()
        b3  = eng.allocate_qubit()

        b4  = eng.allocate_qubit()
        b5  = eng.allocate_qubit()
        b6  = eng.allocate_qubit()
        b7  = eng.allocate_qubit()

        b8  = eng.allocate_qubit()
        b9  = eng.allocate_qubit()
        b10 = eng.allocate_qubit()
        b11 = eng.allocate_qubit()

        c0  = eng.allocate_qubit()
        c1  = eng.allocate_qubit()
        c2  = eng.allocate_qubit()
        c3  = eng.allocate_qubit()
```

```c
#include <stdio.h>

void Toffoli(int entity1, int entity2, int *result);
void CNOT(int left, int *right);

/*

Classic McEliece, NTS-KEM

P = x12 + x3 + 1

*/


int main() {

    int a[12];
    int b[12];
    int c[12];

    a[0] = 0;
    a[1] = 0;
    a[2] = 0;
    a[3] = 0;
    a[4] = 0;
    a[5] = 0;
    a[6] = 0;
    a[7] = 0;
    a[8] = 0;
    a[9] = 0;
    a[10] = 0;
    a[11] = 1;

    b[0] = 0;
    b[1] = 1;
    b[2] = 0;
    b[3] = 0;
    b[4] = 0;
    b[5] = 0;
    b[6] = 0;
    b[7] = 0;
```

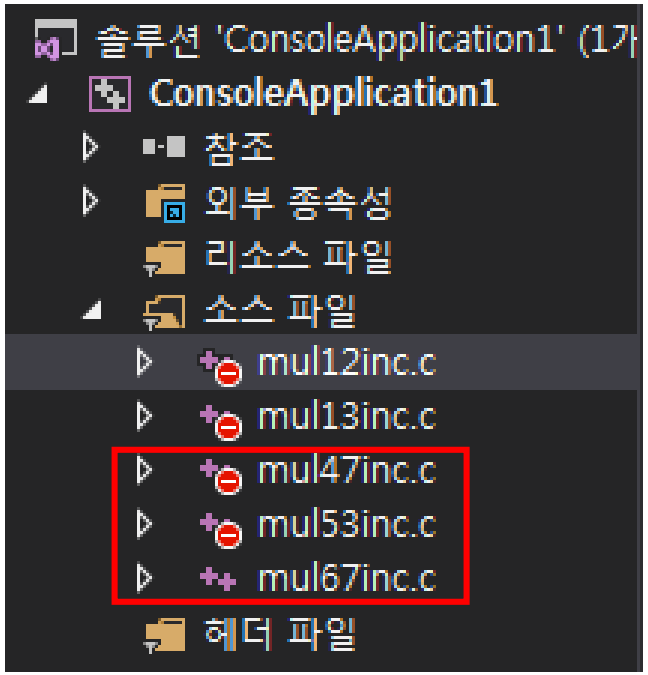
ROLLO 구현

3.3 Parameters

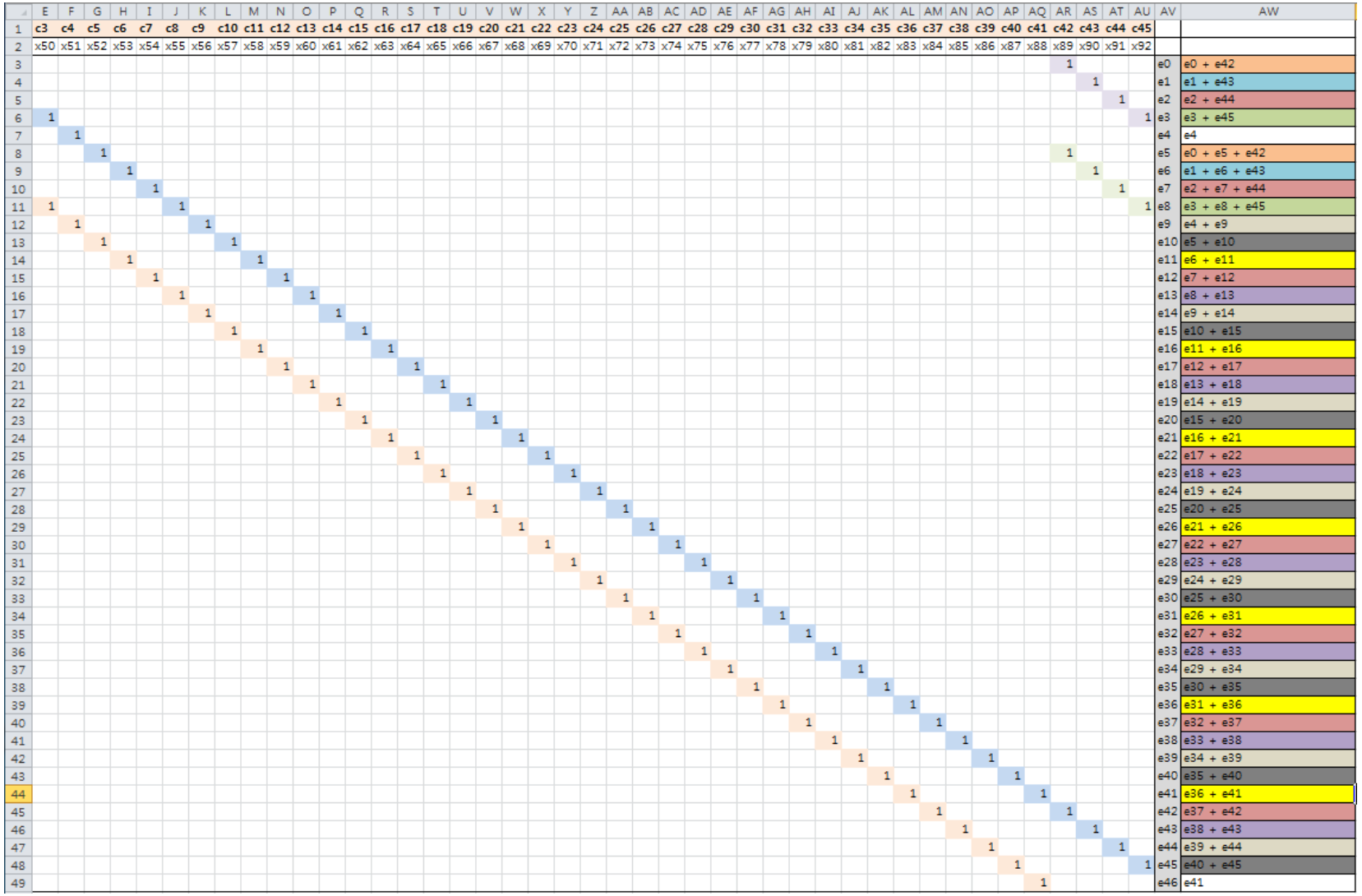
The submission of ROLLO-I allows three different levels of security achieving 128, 192 and 256 bits of security according respectively to NIST's security strength categories 1, 3 and 5 [21], we recall them in Table 3. As described in Section 3, the parameters n and m correspond respectively to the degrees of irreducible polynomials P and P_m used to construct the field $\mathbb{F}_{q^m}^n$ and the parameters d and r correspond respectively to the private key and the error's rank.

Algo. \ Param.	n	m	d	r	P	P_m	Security level (bits)
ROLLO-I-128	47	79	6	5	$X^{47}+X^5+1$	$X^{79}+X^9+1$	128
ROLLO-I-192	53	89	7	6	$X^{53}+X^6+X^2+X+1$	$X^{89}+X^{38}+1$	192
ROLLO-I-256	67	113	8	7	$X^{67}+X^5+X^2+X+1$	$X^{113}+X^9+1$	256

Table 3. ROLLO-I parameters for each security level

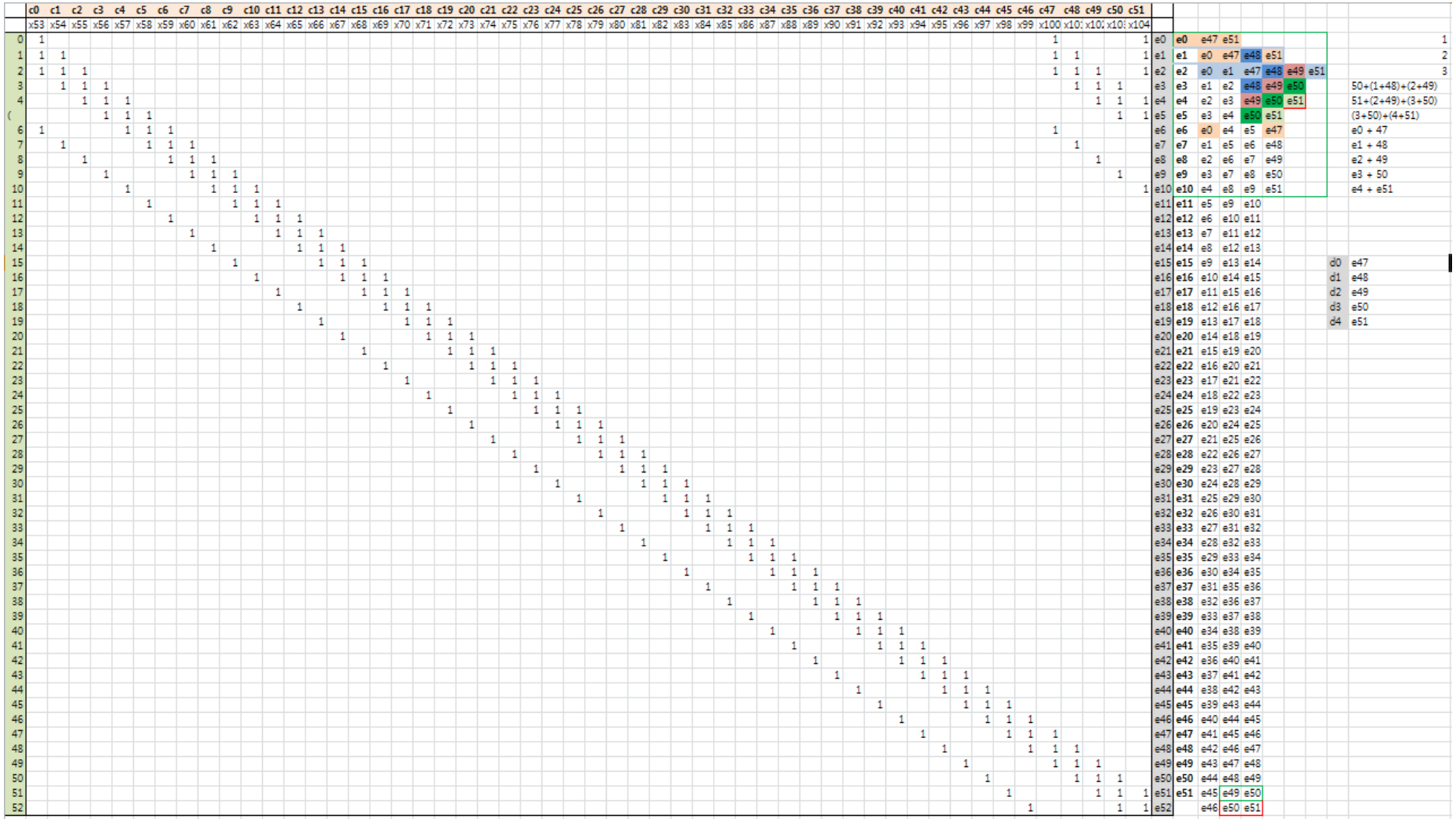


ROLLO x47 + x5 + 1



	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW
1	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19	c20	c21	c22	c23	c24	c25	c26	c27	c28	c29	c30	c31	c32	c33	c34	c35	c36	c37	c38	c39	c40	c41	c42	c43	c44	c45		
2	x50	x51	x52	x53	x54	x55	x56	x57	x58	x59	x60	x61	x62	x63	x64	x65	x66	x67	x68	x69	x70	x71	x72	x73	x74	x75	x76	x77	x78	x79	x80	x81	x82	x83	x84	x85	x86	x87	x88	x89	x90	x91	x92		
3																																								1				e0	e0 + e42
4																																									1			e1	e1 + e43
5																																										1		e2	e2 + e44
6	1																																										1	e3	e3 + e45
7		1																																										e4	e4
8			1																																						1			e5	e0 + e5 + e42
9				1																																					1			e6	e1 + e6 + e43
10					1																																					1		e7	e2 + e7 + e44
11	1					1																																					1	e8	e3 + e8 + e45
12		1					1																																					e9	e4 + e9
13			1					1																																				e10	e5 + e10
14				1					1																																			e11	e6 + e11
15					1					1																																			e12	e7 + e12
16						1					1																																		e13	e8 + e13
17							1					1																																	e14	e9 + e14
18								1					1																																e15	e10 + e15
19									1					1																															e16	e11 + e16
20										1					1																														e17	e12 + e17
21											1					1																													e18	e13 + e18
22												1					1																												e19	e14 + e19
23													1					1																											e20	e15 + e20
24														1					1																										e21	e16 + e21
25															1					1																									e22	e17 + e22
26																1					1																								e23	e18 + e23
27																	1					1																							e24	e19 + e24
28																		1					1																						e25	e20 + e25
29																			1					1																					e26	e21 + e26
30																				1					1																				e27	e22 + e27
31																					1					1																			e28	e23 + e28
32																						1					1																		e29	e24 + e29
33																							1					1																	e30	e25 + e30
34																								1					1																e31	e26 + e31
35																									1					1															e32	e27 + e32
36																										1					1														e33	e28 + e33
37																											1					1													e34	e29 + e34
38																												1					1												e35	e30 + e35
39																													1					1											e36	e31 + e36
40																														1					1										e37	e32 + e37
41																															1					1									e38	e33 + e38
42																																1					1								e39	e34 + e39
43																																	1					1							e40	e35 + e40
44																																		1					1						e41	e36 + e41
45																																			1					1					e42	e37 + e42
46																																				1					1				e43	e38 + e43
47																																					1					1			e44	e39 + e44
48																																						1					1		e45	e40 + e45
49																																							1						e46	e41

ROLLO x53 + x6 + x2 + x1 + 1



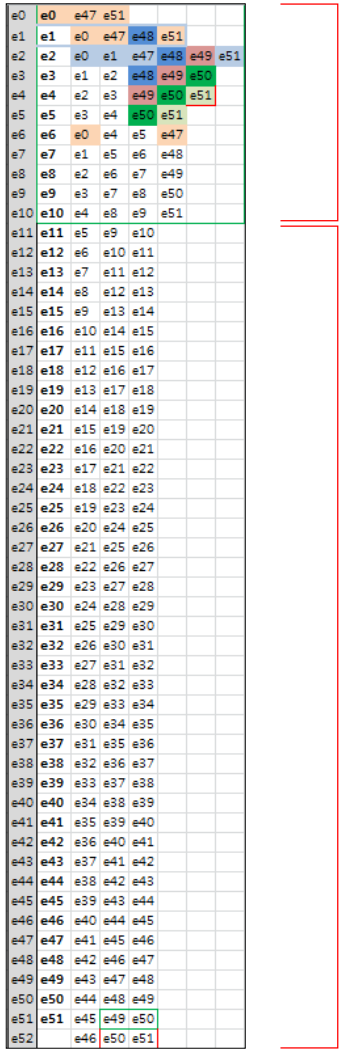
X53 + x6 + x2 + x1 + 1

ROLLO x53 + x6 + x2 + x1 + 1

e0	e0 + e42
e1	e1 + e43
e2	e2 + e44
e3	e3 + e45
e4	e4
e5	e0 + e5 + e42
e6	e1 + e6 + e43
e7	e2 + e7 + e44
e8	e3 + e8 + e45
e9	e4 + e9
e10	e5 + e10
e11	e6 + e11
e12	e7 + e12
e13	e8 + e13
e14	e9 + e14
e15	e10 + e15
e16	e11 + e16
e17	e12 + e17
e18	e13 + e18
e19	e14 + e19
e20	e15 + e20
e21	e16 + e21
e22	e17 + e22
e23	e18 + e23
e24	e19 + e24
e25	e20 + e25
e26	e21 + e26
e27	e22 + e27
e28	e23 + e28
e29	e24 + e29
e30	e25 + e30
e31	e26 + e31
e32	e27 + e32
e33	e28 + e33
e34	e29 + e34
e35	e30 + e35
e36	e31 + e36
e37	e32 + e37
e38	e33 + e38
e39	e34 + e39
e40	e35 + e40
e41	e36 + e41
e42	e37 + e42
e43	e38 + e43
e44	e39 + e44
e45	e40 + e45
e46	e41

e0	**e0**	e47	e51						1
e1	**e1**	e0	e47	e48	e51				2
e2	**e2**	e0	e1	e47	e48	e49	e51		3
e3	**e3**	e1	e2	e48	e49	e50			50+(1+48)+(2+49)
e4	**e4**	e2	e3	e49	e50	e51			51+(2+49)+(3+50)
e5	**e5**	e3	e4	e50	e51				(3+50)+(4+51)
e6	**e6**	e0	e4	e5	e47				e0 + 47
e7	**e7**	e1	e5	e6	e48				e1 + 48
e8	**e8**	e2	e6	e7	e49				e2 + 49
e9	**e9**	e3	e7	e8	e50				e3 + 50
e10	**e10**	e4	e8	e9	e51				e4 + e51
e11	**e11**	e5	e9	e10					
e12	**e12**	e6	e10	e11					
e13	**e13**	e7	e11	e12					
e14	**e14**	e8	e12	e13					
e15	**e15**	e9	e13	e14				d0	e47
e16	**e16**	e10	e14	e15				d1	e48
e17	**e17**	e11	e15	e16				d2	e49
e18	**e18**	e12	e16	e17				d3	e50
e19	**e19**	e13	e17	e18				d4	e51
e20	**e20**	e14	e18	e19					
e21	**e21**	e15	e19	e20					
e22	**e22**	e16	e20	e21					
e23	**e23**	e17	e21	e22					
e24	**e24**	e18	e22	e23					
e25	**e25**	e19	e23	e24					
e26	**e26**	e20	e24	e25					
e27	**e27**	e21	e25	e26					
e28	**e28**	e22	e26	e27					
e29	**e29**	e23	e27	e28					
e30	**e30**	e24	e28	e29					
e31	**e31**	e25	e29	e30					
e32	**e32**	e26	e30	e31					
e33	**e33**	e27	e31	e32					
e34	**e34**	e28	e32	e33					
e35	**e35**	e29	e33	e34					
e36	**e36**	e30	e34	e35					
e37	**e37**	e31	e35	e36					
e38	**e38**	e32	e36	e37					
e39	**e39**	e33	e37	e38					
e40	**e40**	e34	e38	e39					
e41	**e41**	e35	e39	e40					
e42	**e42**	e36	e40	e41					
e43	**e43**	e37	e41	e42					
e44	**e44**	e38	e42	e43					
e45	**e45**	e39	e43	e44					
e46	**e46**	e40	e44	e45					
e47	**e47**	e41	e45	e46					
e48	**e48**	e42	e46	e47					
e49	**e49**	e43	e47	e48					
e50	**e50**	e44	e48	e49					
e51	**e51**	e45	e49	e50					
e52		e46	e50	e51					

ROLLO x53 + x6 + x2 + x1 + 1



e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		
e11	**e11**	e5	e9	e10			
e12	**e12**	e6	e10	e11			
e13	**e13**	e7	e11	e12			
e14	**e14**	e8	e12	e13			
e15	**e15**	e9	e13	e14			
e16	**e16**	e10	e14	e15			
e17	**e17**	e11	e15	e16			
e18	**e18**	e12	e16	e17			
e19	**e19**	e13	e17	e18			
e20	**e20**	e14	e18	e19			
e21	**e21**	e15	e19	e20			
e22	**e22**	e16	e20	e21			
e23	**e23**	e17	e21	e22			
e24	**e24**	e18	e22	e23			
e25	**e25**	e19	e23	e24			
e26	**e26**	e20	e24	e25			
e27	**e27**	e21	e25	e26			
e28	**e28**	e22	e26	e27			
e29	**e29**	e23	e27	e28			
e30	**e30**	e24	e28	e29			
e31	**e31**	e25	e29	e30			
e32	**e32**	e26	e30	e31			
e33	**e33**	e27	e31	e32			
e34	**e34**	e28	e32	e33			
e35	**e35**	e29	e33	e34			
e36	**e36**	e30	e34	e35			
e37	**e37**	e31	e35	e36			
e38	**e38**	e32	e36	e37			
e39	**e39**	e33	e37	e38			
e40	**e40**	e34	e38	e39			
e41	**e41**	e35	e39	e40			
e42	**e42**	e36	e40	e41			
e43	**e43**	e37	e41	e42			
e44	**e44**	e38	e42	e43			
e45	**e45**	e39	e43	e44			
e46	**e46**	e40	e44	e45			
e47	**e47**	e41	e45	e46			
e48	**e48**	e42	e46	e47			
e49	**e49**	e43	e47	e48			
e50	**e50**	e44	e48	e49			
e51	**e51**	e45	e49	e50			
e52		e46	e50	e51			

e52 → e11

- 만약 e11 → e52 순으로 하는 경우

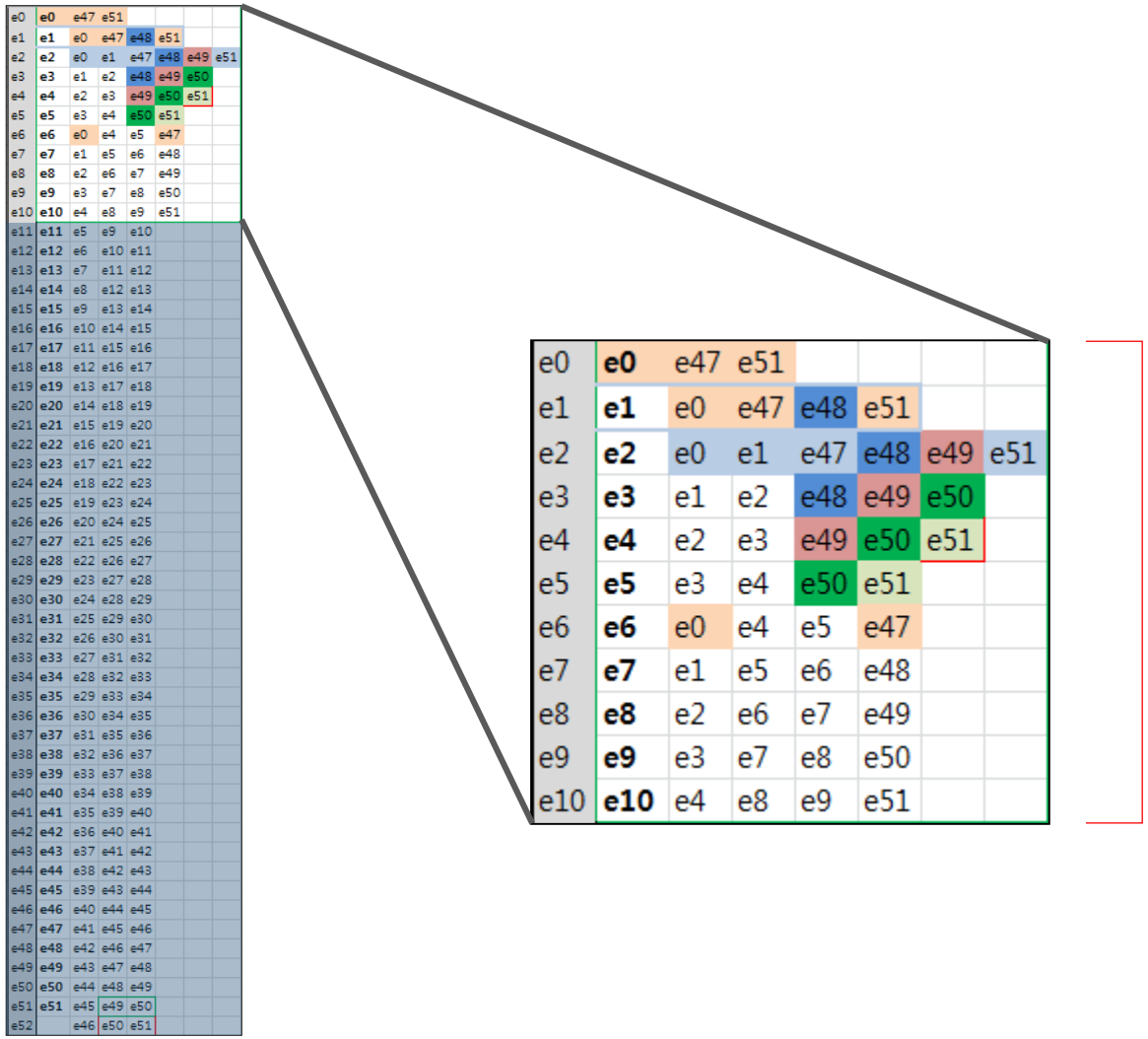
  ex)

  e11 = e5 + e9 + e10

  e12 = e6 + e10 + e11

  e12 = e6 + e10 + (e5 + e9 + e10) X

ROLLO x53 + x6 + x2 + x1 + 1



e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

연산 횟수 최적화 가능 부분

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e4 = e2 + e3 + e49 + e50 +e51
  - e4 = + e51
  - e10 = e4

    e10 = e4 + e51

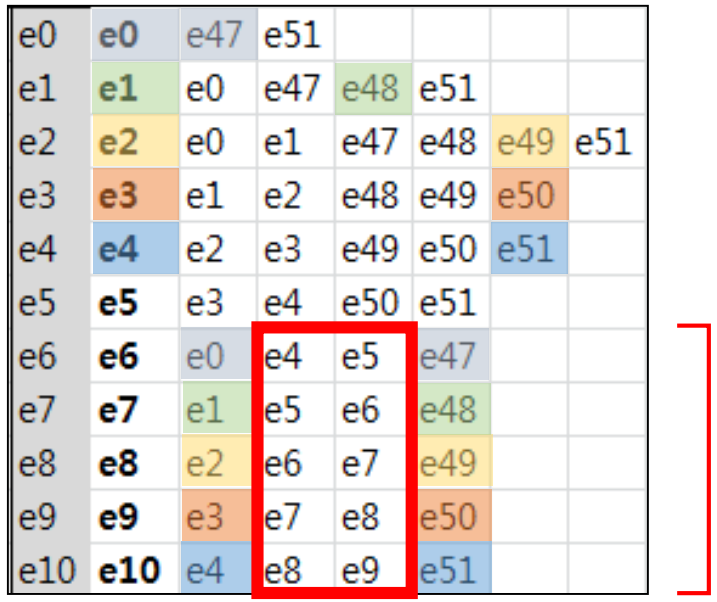
    e10 = e4 + e51 + e8 + e9

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e6, e7, e8 , e9, e10
  - 전부 다 적용 가능
  - 연산 횟수 4회 → 3회

ROLLO x53 + x6 + x2 + x1 + 1



e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e6, e7, e8 , e9, e10
  - 전부 다 적용 가능
  - 연산 횟수 4회 → 3회

- e6, e7, e8 , e9, e10

  ex)

  e6 = e0(e0+e47)

  e6 = e0 + e47 + e4 + e5

  …

  e8 = e2(e2 + e49) + e6 + e7

  e8 = e2 + e49 + (e0 + e47 + e4 + e5) + e7 X

무조건 안쪽 값부터 연산

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e10 = e9 + e8
- e 9 = e8 + e7
- e8 = e7 + e6
- e7 = e6 + e5
- e6 = e5 + e4

- e0 = e0 + e47
- e1 = e1 + e48
- e2 = e2 + e49
- e3 = e3 + e50
- e4 = e4 + e51

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e10 = e9 + e8
- e 9 = e8 + e7
- e8 = e7 + e6
- e7 = e6 + e5
- e6 = e5 + e4

- e0 = e0 + e47
- e1 = e1 + e48
- e2 = e2 + e49
- e3 = e3 + e50
- e4 = e4 + e51

- e10 = e10 + e4(e4 + e51)
- e 9 = e9 + e3(e3 + e50)
- e8 = e8 + e2(e2 + e49)
- e7 = e7 + e1(e1 + e48)
- e6 = e6 + e0(e0 + e47)

e6, e7, e8, e9, e10 완료

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e5 = e3(e3 + e50) + e4(e4 + e51)

  e5 완료

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e4 = e2(e2 + e49) + e3(e3 + e50) + e51

  e4 완료

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e3 = e1(e1 + e48) + e2(e2 + e49) + e50

  e3 완료

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

- e3 = e1(e1 + e48) + e2(e2 + e49)

  e3 완료

ROLLO x53 + x6 + x2 + x1 + 1

e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

1. e0 = e0 + **e51** ➔ **e0 완성**
2. e1 = e1 + e0(e0 + e47 + e51) = e1 + e0 + e47 + e48 + e51
   ➔ **e1 완성**

ROLLO x53 + x6 + x2 + x1 + 1

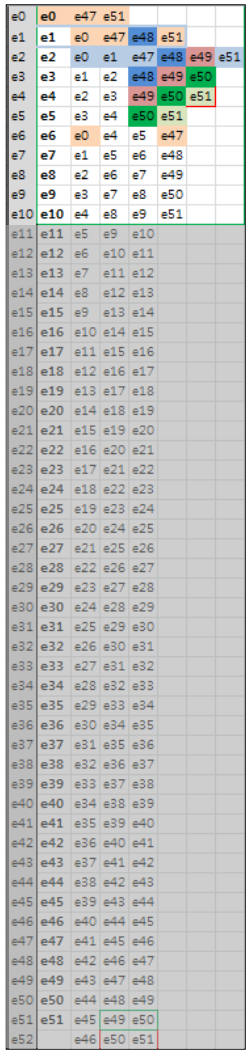
e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

1. e0 = e0 + **e51** ➔ e0 완성
2. e1 = e1 + e0(e0 + e47 + e51) = e1 + e0 + e47 + e48 + e51
   ➔ **e1 완성**

3. e2 = e2 + e1(e0 + e1 + e47 + e48 + e51) = e2 + e0 + e1 + e47 + e48 + e49 + e51
   ➔ **e2 완성**

전부 완성

ROLLO x53 + x6 + x2 + x1 + 1



e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

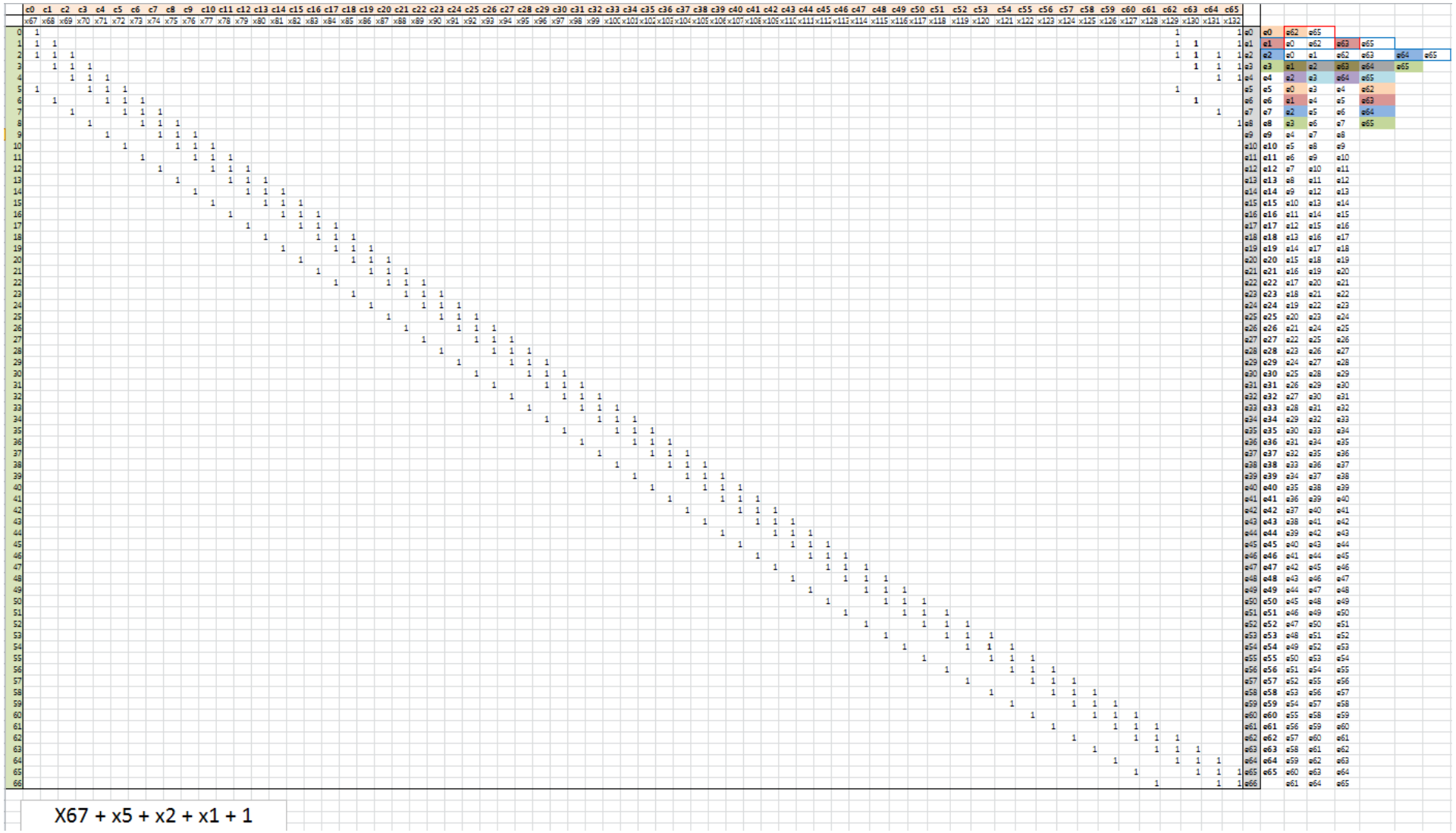
연산 되기 전의 e47~e51 값을

저장할 공간 따로 필요(5개)

e47~e51 값들이 변경

ex) e47 = e41 + e45 + e46

ROLLO x67 + x5 + x2 + x1 + 1



X67 + x5 + x2 + x1 + 1

ROLLO 구현

```
15  int main() {
16
17      int a[12];
18      int b[12];
19      int c[12];
20
21      a[0] = 0;
22      a[1] = 0;
23      a[2] = 0;
24      a[3] = 0;
25      a[4] = 0;
26      a[5] = 0;
27      a[6] = 0;
28      a[7] = 0;
29      a[8] = 0;
30      a[9] = 0;
31      a[10] = 0;
32      a[11] = 1;
33
34      b[0] = 0;
35      b[1] = 1;
36      b[2] = 0;
37      b[3] = 0;
38      b[4] = 0;
39      b[5] = 0;
40      b[6] = 0;
41      b[7] = 0;
42      b[8] = 0;
43      b[9] = 0;
44      b[10] = 0;
45      b[11] = 0;
46
47      c[0] = 0;
48      c[1] = 0;
49      c[2] = 0;
50      c[3] = 0;
51      c[4] = 0;
52      c[5] = 0;
53      c[6] = 0;
54      c[7] = 0;
55      c[8] = 0;
```

변수 선언 및 초기화

```
93
94
95      Toffoli(a[11], b[1], &c[0]);
96      Toffoli(a[10], b[2], &c[0]);
97      Toffoli(a[9], b[3], &c[0]);
98      Toffoli(a[8], b[4], &c[0]);
99      Toffoli(a[7], b[5], &c[0]);
100     Toffoli(a[6], b[6], &c[0]);
101     Toffoli(a[5], b[7], &c[0]);
102     Toffoli(a[4], b[8], &c[0]);
103     Toffoli(a[3], b[9], &c[0]);
104     Toffoli(a[2], b[10], &c[0]);
105     Toffoli(a[1], b[11], &c[0]);//11
106
107
108     Toffoli(a[11], b[2], &c[1]);
109     Toffoli(a[10], b[3], &c[1]);
110     Toffoli(a[9], b[4], &c[1]);
111     Toffoli(a[8], b[5], &c[1]);
112     Toffoli(a[7], b[6], &c[1]);
113     Toffoli(a[6], b[7], &c[1]);
114     Toffoli(a[5], b[8], &c[1]);
115     Toffoli(a[4], b[9], &c[1]);
116     Toffoli(a[3], b[10], &c[1]);
117     Toffoli(a[2], b[11], &c[1]);//10
118
119
120     Toffoli(a[11], b[3], &c[2]);
121     Toffoli(a[10], b[4], &c[2]);
122     Toffoli(a[9], b[5], &c[2]);
123     Toffoli(a[8], b[6], &c[2]);
124     Toffoli(a[7], b[7], &c[2]);
125     Toffoli(a[6], b[8], &c[2]);
126     Toffoli(a[5], b[9], &c[2]);
127     Toffoli(a[4], b[10], &c[2]);
128     Toffoli(a[3], b[11], &c[2]);
129
130
131     Toffoli(a[11], b[4], &c[3]);
132     Toffoli(a[10], b[5], &c[3]);
```

Reduction이 필요한 곱셈 연산

```
184     ////MIXING
185
186
187     CNOT(c[8], &c[11]);
188     CNOT(c[5], &c[8]);
189
190     CNOT(c[9], &c[0]);
191     CNOT(c[10], &c[1]);
192
193     CNOT(c[6], &c[9]);
194     CNOT(c[7], &c[10]);
195
196     CNOT(c[2], &c[5]);
197     CNOT(c[3], &c[6]);
198     CNOT(c[4], &c[7]);
199
200     CNOT(c[0], &c[3]);
201     CNOT(c[1], &c[4]);
202
```

앞 곱셈 연산에 대한 Reduction

```
217
218     Toffoli(a[0], b[0], &c[0]);
219
220     Toffoli(a[1], b[0], &c[1]);
221     Toffoli(a[0], b[1], &c[1]);
222
223     Toffoli(a[2], b[0], &c[2]);
224     Toffoli(a[1], b[1], &c[2]);
225     Toffoli(a[0], b[2], &c[2]);
226
227     Toffoli(a[3], b[0], &c[3]);
228     Toffoli(a[2], b[1], &c[3]);
229     Toffoli(a[1], b[2], &c[3]);
230     Toffoli(a[0], b[3], &c[3]);
231
232     Toffoli(a[4], b[0], &c[4]);
233     Toffoli(a[3], b[1], &c[4]);
234     Toffoli(a[2], b[2], &c[4]);
235     Toffoli(a[1], b[3], &c[4]);
236     Toffoli(a[0], b[4], &c[4]);
237
238     Toffoli(a[5], b[0], &c[5]);
239     Toffoli(a[4], b[1], &c[5]);
240     Toffoli(a[3], b[2], &c[5]);
241     Toffoli(a[2], b[3], &c[5]);
242     Toffoli(a[1], b[4], &c[5]);
243     Toffoli(a[0], b[5], &c[5]);
244
245     Toffoli(a[6], b[0], &c[6]);
246     Toffoli(a[5], b[1], &c[6]);
247     Toffoli(a[4], b[2], &c[6]);
248     Toffoli(a[3], b[3], &c[6]);
249     Toffoli(a[2], b[4], &c[6]);
250     Toffoli(a[1], b[5], &c[6]);
251     Toffoli(a[0], b[6], &c[6]);
252
253     Toffoli(a[7], b[0], &c[7]);
254     Toffoli(a[6], b[1], &c[7]);
```

Reduction이 필요 없는 곱셈 연산

ROLLO 구현

```
int main() {

    int a[12];
    int b[12];
    int c[12];

    a[0] = 0;
    a[1] = 0;
    a[2] = 0;
    a[3] = 0;
    a[4] = 0;
    a[5] = 0;
    a[6] = 0;
    a[7] = 0;
    a[8] = 0;
    a[9] = 0;
    a[10] = 0;
    a[11] = 1;

    b[0] = 0;
    b[1] = 1;
    b[2] = 0;
    b[3] = 0;
    b[4] = 0;
    b[5] = 0;
    b[6] = 0;
    b[7] = 0;
    b[8] = 0;
    b[9] = 0;
    b[10] = 0;
    b[11] = 0;

    c[0] = 0;
    c[1] = 0;
    c[2] = 0;
    c[3] = 0;
    c[4] = 0;
    c[5] = 0;
    c[6] = 0;
    c[7] = 0;
    c[8] = 0;
```

변수 선언 및 초기화

Example x12 + x3 + 1

```

    Toffoli(a[11], b[1], &c[0]);
    Toffoli(a[10], b[2], &c[0]);
    Toffoli(a[9], b[3], &c[0]);
    Toffoli(a[8], b[4], &c[0]);
    Toffoli(a[7], b[5], &c[0]);
    Toffoli(a[6], b[6], &c[0]);
    Toffoli(a[5], b[7], &c[0]);
    Toffoli(a[4], b[8], &c[0]);
    Toffoli(a[3], b[9], &c[0]);
    Toffoli(a[2], b[10], &c[0]);
    Toffoli(a[1], b[11], &c[0]);//11

    Toffoli(a[11], b[2], &c[1]);
    Toffoli(a[10], b[3], &c[1]);
    Toffoli(a[9], b[4], &c[1]);
    Toffoli(a[8], b[5], &c[1]);
    Toffoli(a[7], b[6], &c[1]);
    Toffoli(a[6], b[7], &c[1]);
    Toffoli(a[5], b[8], &c[1]);
    Toffoli(a[4], b[9], &c[1]);
    Toffoli(a[3], b[10], &c[1]);
    Toffoli(a[2], b[11], &c[1]);//10

    Toffoli(a[11], b[3], &c[2]);
    Toffoli(a[10], b[4], &c[2]);
    Toffoli(a[9], b[5], &c[2]);
    Toffoli(a[8], b[6], &c[2]);
    Toffoli(a[7], b[7], &c[2]);
    Toffoli(a[6], b[8], &c[2]);
    Toffoli(a[5], b[9], &c[2]);
    Toffoli(a[4], b[10], &c[2]);
    Toffoli(a[3], b[11], &c[2]);

    Toffoli(a[11], b[4], &c[3]);
    Toffoli(a[10], b[5], &c[3]);
```

Reduction이 필요한 곱셈 연산

x12 ~ x24

```
    ////MIXING


    CNOT(c[8], &c[11]);
    CNOT(c[5], &c[8]);

    CNOT(c[9], &c[0]);
    CNOT(c[10], &c[1]);

    CNOT(c[6], &c[9]);
    CNOT(c[7], &c[10]);

    CNOT(c[2], &c[5]);
    CNOT(c[3], &c[6]);
    CNOT(c[4], &c[7]);

    CNOT(c[0], &c[3]);
    CNOT(c[1], &c[4]);
```

앞 곱셈 연산에 대한 Reduction

```

    Toffoli(a[0], b[0], &c[0]);

    Toffoli(a[1], b[0], &c[1]);
    Toffoli(a[0], b[1], &c[1]);

    Toffoli(a[2], b[0], &c[2]);
    Toffoli(a[1], b[1], &c[2]);
    Toffoli(a[0], b[2], &c[2]);

    Toffoli(a[3], b[0], &c[3]);
    Toffoli(a[2], b[1], &c[3]);
    Toffoli(a[1], b[2], &c[3]);
    Toffoli(a[0], b[3], &c[3]);

    Toffoli(a[4], b[0], &c[4]);
    Toffoli(a[3], b[1], &c[4]);
    Toffoli(a[2], b[2], &c[4]);
    Toffoli(a[1], b[3], &c[4]);
    Toffoli(a[0], b[4], &c[4]);

    Toffoli(a[5], b[0], &c[5]);
    Toffoli(a[4], b[1], &c[5]);
    Toffoli(a[3], b[2], &c[5]);
    Toffoli(a[2], b[3], &c[5]);
    Toffoli(a[1], b[4], &c[5]);
    Toffoli(a[0], b[5], &c[5]);

    Toffoli(a[6], b[0], &c[6]);
    Toffoli(a[5], b[1], &c[6]);
    Toffoli(a[4], b[2], &c[6]);
    Toffoli(a[3], b[3], &c[6]);
    Toffoli(a[2], b[4], &c[6]);
    Toffoli(a[1], b[5], &c[6]);
    Toffoli(a[0], b[6], &c[6]);

    Toffoli(a[7], b[0], &c[7]);
    Toffoli(a[6], b[1], &c[7]);
```

Reduction이 필요 없는 곱셈 연산

1 ~ x11

ROLLO 구현

변수 선언 및 초기화

```
int main() {

    int a[12];
    int b[12];
    int c[12];

    a[0] = 0;
    a[1] = 0;
    a[2] = 0;
    a[3] = 0;
    a[4] = 0;
    a[5] = 0;
    a[6] = 0;
    a[7] = 0;
    a[8] = 0;
    a[9] = 0;
    a[10] = 0;
    a[11] = 1;

    b[0] = 0;
    b[1] = 1;
    b[2] = 0;
    b[3] = 0;
    b[4] = 0;
    b[5] = 0;
    b[6] = 0;
    b[7] = 0;
    b[8] = 0;
    b[9] = 0;
    b[10] = 0;
    b[11] = 0;

    c[0] = 0;
    c[1] = 0;
    c[2] = 0;
    c[3] = 0;
    c[4] = 0;
    c[5] = 0;
    c[6] = 0;
    c[7] = 0;
    c[8] = 0;
```

```
int main() {

    int a[53];
    int b[53];
    int c[53];

    int d[5];

    int mulNum = 53;

    for (int i = mulNum - 1; i > -1; i--) {
        a[i] = 0;
        b[i] = 0;
        c[i] = 0;
    }

    // mul 13 때와 마찬가지로 변하지 않은 값을 따로 저장할 공간이 필요함

    d[0] = 0; //c[47]
    d[1] = 0; //c[48]
    d[2] = 0; //c[49]
    d[3] = 0; //c[50]
    d[4] = 0; //c[51]


    // 입력하고 싶은 값 입력하는 곳
    // a,b,c = 0~52
    // ex) a[52] = 1;

    a[52] = 1;
    b[52] = 1; // (1) * (x)
```

ROLLO 구현

Reduction이 필요한 곱셈 연산

Example x12 + x3 + 1

```
93
94
95      Toffoli(a[11], b[1], &c[0]);
96      Toffoli(a[10], b[2], &c[0]);
97      Toffoli(a[9], b[3], &c[0]);
98      Toffoli(a[8], b[4], &c[0]);
99      Toffoli(a[7], b[5], &c[0]);
100     Toffoli(a[6], b[6], &c[0]);
101     Toffoli(a[5], b[7], &c[0]);
102     Toffoli(a[4], b[8], &c[0]);
103     Toffoli(a[3], b[9], &c[0]);
104     Toffoli(a[2], b[10], &c[0]);
105     Toffoli(a[1], b[11], &c[0]);//11
106
107
108     Toffoli(a[11], b[2], &c[1]);
109     Toffoli(a[10], b[3], &c[1]);
110     Toffoli(a[9], b[4], &c[1]);
111     Toffoli(a[8], b[5], &c[1]);
112     Toffoli(a[7], b[6], &c[1]);
113     Toffoli(a[6], b[7], &c[1]);
114     Toffoli(a[5], b[8], &c[1]);
115     Toffoli(a[4], b[9], &c[1]);
116     Toffoli(a[3], b[10], &c[1]);
117     Toffoli(a[2], b[11], &c[1]);//10
118
119
120     Toffoli(a[11], b[3], &c[2]);
121     Toffoli(a[10], b[4], &c[2]);
122     Toffoli(a[9], b[5], &c[2]);
123     Toffoli(a[8], b[6], &c[2]);
124     Toffoli(a[7], b[7], &c[2]);
125     Toffoli(a[6], b[8], &c[2]);
126     Toffoli(a[5], b[9], &c[2]);
127     Toffoli(a[4], b[10], &c[2]);
128     Toffoli(a[3], b[11], &c[2]);
129
130
131     Toffoli(a[11], b[4], &c[3]);
132     Toffoli(a[10], b[5], &c[3]);
```

11번

10번

9번

…

1번

- X67 + x5 + x2 + x + 1

66번

65번

64번

63번

…

3번

2번

1번

ROLLO 구현

Reduction이 필요한 곱셈 연산

Example x12 + x3 + 1

```
93
94
95      Toffoli(a[11], b[1], &c[0]);
96      Toffoli(a[10], b[2], &c[0]);
97      Toffoli(a[9], b[3], &c[0]);
98      Toffoli(a[8], b[4], &c[0]);
99      Toffoli(a[7], b[5], &c[0]);
100     Toffoli(a[6], b[6], &c[0]);
101     Toffoli(a[5], b[7], &c[0]);
102     Toffoli(a[4], b[8], &c[0]);
103     Toffoli(a[3], b[9], &c[0]);
104     Toffoli(a[2], b[10], &c[0]);
105     Toffoli(a[1], b[11], &c[0]);//11
106
107
108     Toffoli(a[11], b[2], &c[1]);
109     Toffoli(a[10], b[3], &c[1]);
110     Toffoli(a[9], b[4], &c[1]);
111     Toffoli(a[8], b[5], &c[1]);
112     Toffoli(a[7], b[6], &c[1]);
113     Toffoli(a[6], b[7], &c[1]);
114     Toffoli(a[5], b[8], &c[1]);
115     Toffoli(a[4], b[9], &c[1]);
116     Toffoli(a[3], b[10], &c[1]);
117     Toffoli(a[2], b[11], &c[1]);//10
118
119
120     Toffoli(a[11], b[3], &c[2]);
121     Toffoli(a[10], b[4], &c[2]);
122     Toffoli(a[9], b[5], &c[2]);
123     Toffoli(a[8], b[6], &c[2]);
124     Toffoli(a[7], b[7], &c[2]);
125     Toffoli(a[6], b[8], &c[2]);
126     Toffoli(a[5], b[9], &c[2]);
127     Toffoli(a[4], b[10], &c[2]);
128     Toffoli(a[3], b[11], &c[2]);
129
130
131     Toffoli(a[11], b[4], &c[3]);
132     Toffoli(a[10], b[5], &c[3]);
```

11번

10번

9번

…

1번

- X67 + x5 + x2 + x + 1

66번

65번

64번

63번

…

3번

2번

1번

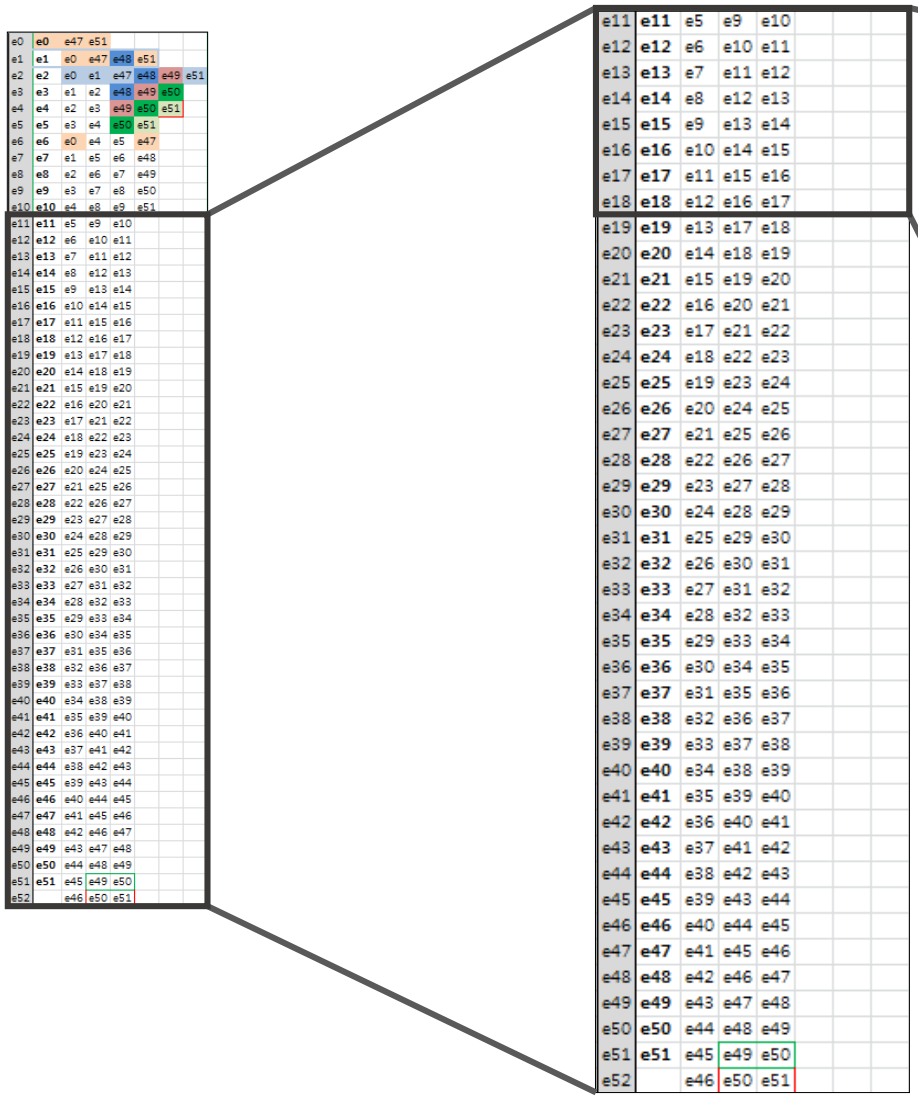
```
int targetNumber = 67;

int calcNum = targetNumber - 1;
int startAdjust = targetNumber - 2;
int adjust = startAdjust;
int limit = 0;

  for (int i = 0; i < calcNum; i++) {
    for (int z = calcNum; z > limit; z--) {
      Toffoli(a[z], b[calcNum - adjust], &c[i]);
      adjust = adjust - 1;
     }
  limit = limit + 1;
  startAdjust = startAdjust - 1;
  adjust = startAdjust;
}
```

ROLLO 구현

앞 곱셈 연산에 대한 Reduction

- x53 + x6 + x2 + x1 + 1

```
184     ////MIXING
185
186
187     CNOT(c[8], &c[11]);
188     CNOT(c[5], &c[8]);
189
190     CNOT(c[9], &c[0]);
191     CNOT(c[10], &c[1]);
192
193     CNOT(c[6], &c[9]);
194     CNOT(c[7], &c[10]);
195
196     CNOT(c[2], &c[5]);
197     CNOT(c[3], &c[6]);
198     CNOT(c[4], &c[7]);
199
200     CNOT(c[0], &c[3]);
201     CNOT(c[1], &c[4]);
202
```

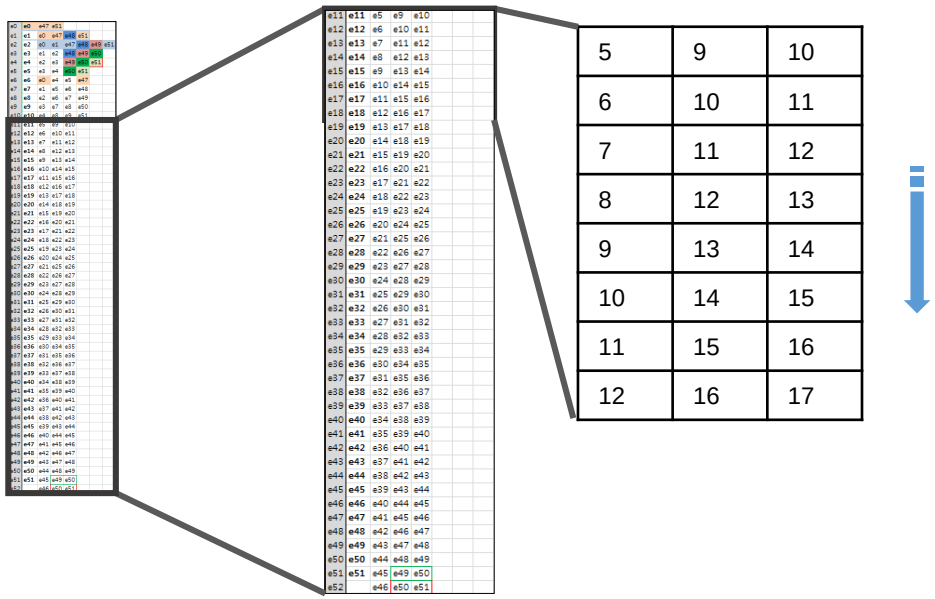


5	9	10
6	10	11
7	11	12
8	12	13
9	13	14
10	14	15
11	15	16
12	16	17

ROLLO 구현

앞 곱셈 연산에 대한 Reduction

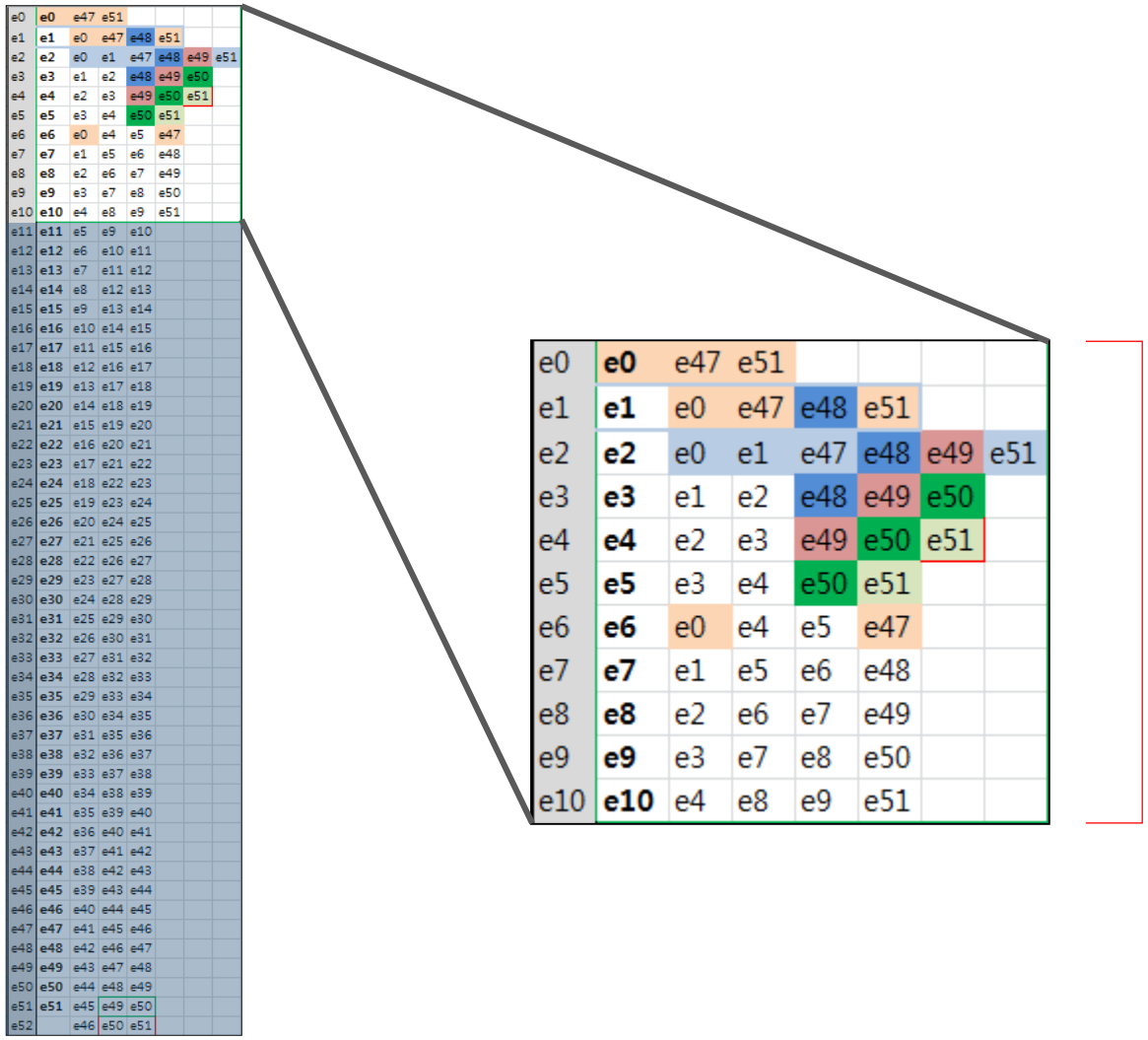
- x53 + x6 + x2 + x1 + 1



5	9	10
6	10	11
7	11	12
8	12	13
9	13	14
10	14	15
11	15	16
12	16	17

```
//e52 ~ e11, 즉 C[52]~C[11]까지 52->11 순으로 연산된 값으로 채워짐
for (int j = 52; j > 10; j--) {
    CNOT(c[j-1], &c[j]);
    CNOT(c[j-2], &c[j]);
    CNOT(c[j-6], &c[j]);
}
```

ROLLO 구현



e0	**e0**	e47	e51				
e1	**e1**	e0	e47	e48	e51		
e2	**e2**	e0	e1	e47	e48	e49	e51
e3	**e3**	e1	e2	e48	e49	e50	
e4	**e4**	e2	e3	e49	e50	e51	
e5	**e5**	e3	e4	e50	e51		
e6	**e6**	e0	e4	e5	e47		
e7	**e7**	e1	e5	e6	e48		
e8	**e8**	e2	e6	e7	e49		
e9	**e9**	e3	e7	e8	e50		
e10	**e10**	e4	e8	e9	e51		

연산 횟수 최적화 가능 부분

ROLLO 구현

```
217
218     Toffoli(a[0], b[0], &c[0]);
219
220     Toffoli(a[1], b[0], &c[1]);
221     Toffoli(a[0], b[1], &c[1]);
222
223     Toffoli(a[2], b[0], &c[2]);
224     Toffoli(a[1], b[1], &c[2]);
225     Toffoli(a[0], b[2], &c[2]);
226
227     Toffoli(a[3], b[0], &c[3]);
228     Toffoli(a[2], b[1], &c[3]);
229     Toffoli(a[1], b[2], &c[3]);
230     Toffoli(a[0], b[3], &c[3]);
231
232     Toffoli(a[4], b[0], &c[4]);
233     Toffoli(a[3], b[1], &c[4]);
234     Toffoli(a[2], b[2], &c[4]);
235     Toffoli(a[1], b[3], &c[4]);
236     Toffoli(a[0], b[4], &c[4]);
237
238     Toffoli(a[5], b[0], &c[5]);
239     Toffoli(a[4], b[1], &c[5]);
240     Toffoli(a[3], b[2], &c[5]);
241     Toffoli(a[2], b[3], &c[5]);
242     Toffoli(a[1], b[4], &c[5]);
243     Toffoli(a[0], b[5], &c[5]);
244
245     Toffoli(a[6], b[0], &c[6]);
246     Toffoli(a[5], b[1], &c[6]);
247     Toffoli(a[4], b[2], &c[6]);
248     Toffoli(a[3], b[3], &c[6]);
249     Toffoli(a[2], b[4], &c[6]);
250     Toffoli(a[1], b[5], &c[6]);
251     Toffoli(a[0], b[6], &c[6]);
252
253     Toffoli(a[7], b[0], &c[7]);
254     Toffoli(a[6], b[1], &c[7]);
```

1회

2회

3회

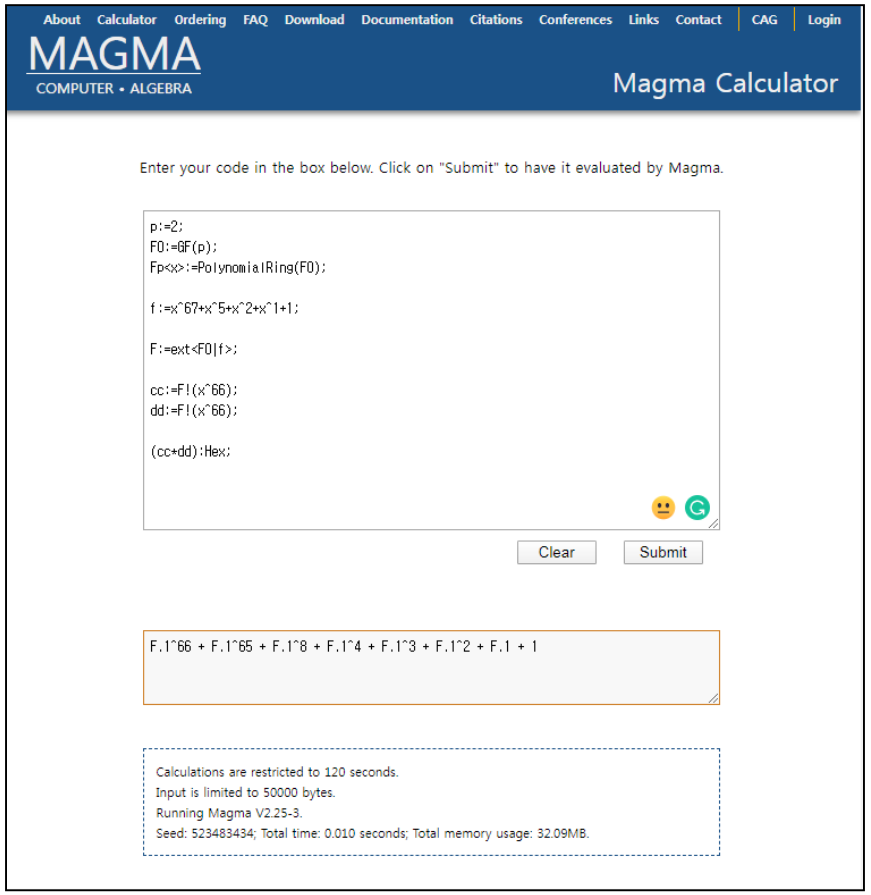
…

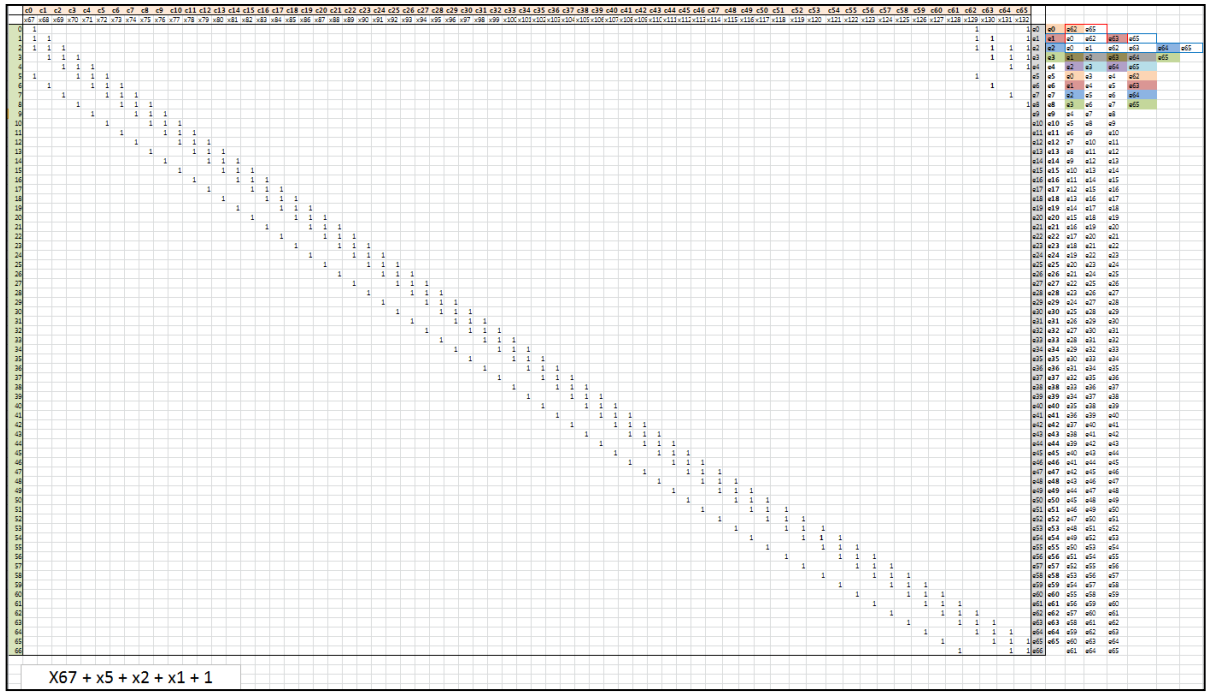
11회

```
////MULTIPLICATION
for (int i = 0; i < targetNumber; i++) {
    for (int z = i; z >= 0; z--) {
        Toffoli(a[z], b[i - z], &c[i]);
    }
}
```

Reduction이 필요 없는 곱셈 연산

ROLLO 구현


About Calculator Ordering FAQ Download Documentation Citations Conferences Links Contact CAG Login
MAGMA
COMPUTER • ALGEBRA
Magma Calculator
Enter your code in the box below. Click on "Submit" to have it evaluated by Magma.
p:=2;
F0:=GF(p);
Fp<x>:=PolynomialRing(F0);
f:=x^67+x^5+x^2+x^1+1;
F:=ext<F0|f>;
cc:=F!(x^66);
dd:=F!(x^66);
(cc+dd):Hex;
Clear
Submit
F.1^66 + F.1^65 + F.1^8 + F.1^4 + F.1^3 + F.1^2 + F.1 + 1
Calculations are restricted to 120 seconds.
Input is limited to 50000 bytes.
Running Magma V2.25-3.
Seed: 523483434; Total time: 0.010 seconds; Total memory usage: 32.09MB.


X67 + x5 + x2 + x1 + 1

ROLLO / RQC

Algo. \ Param.	n	m	d	r	P	P_m	Security level (bits)
ROLLO-I-128	47	79	6	5	$X^{47}+X^5+1$	$X^{79}+X^9+1$	128
ROLLO-I-192	53	89	7	6	$X^{53}+X^6+X^2+X+1$	$X^{89}+X^{38}+1$	192
ROLLO-I-256	67	113	8	7	$X^{67}+X^5+X^2+X+1$	$X^{113}+X^9+1$	256

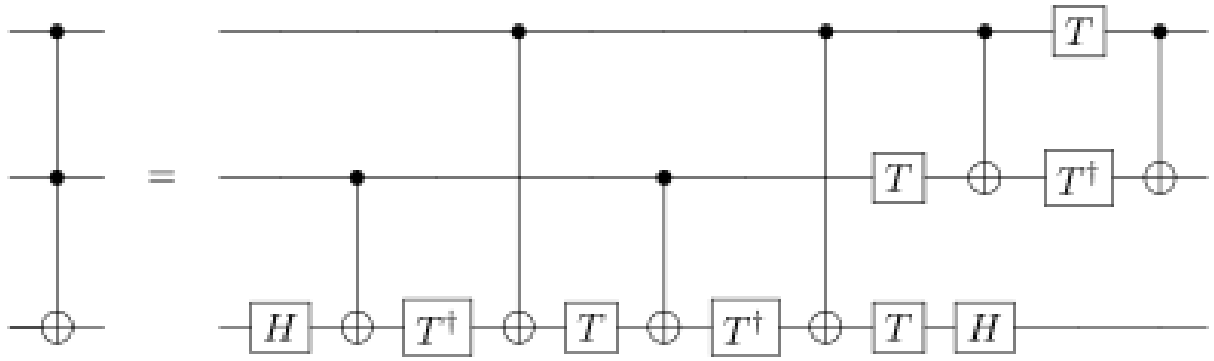
Table 3. ROLLO-I parameters for each security level

Instance	P	Π
RQC-I	$X^{67}+X^5+X^2+X+1$	$X^{97}+X^6+1$
RQC-II	$X^{101}+X^7+X^6+X+1$	$X^{107}+X^9+X^7+X^4+1$
RQC-III	$X^{131}+X^8+X^3+X^2+1$	$X^{137}+X^{21}+1$

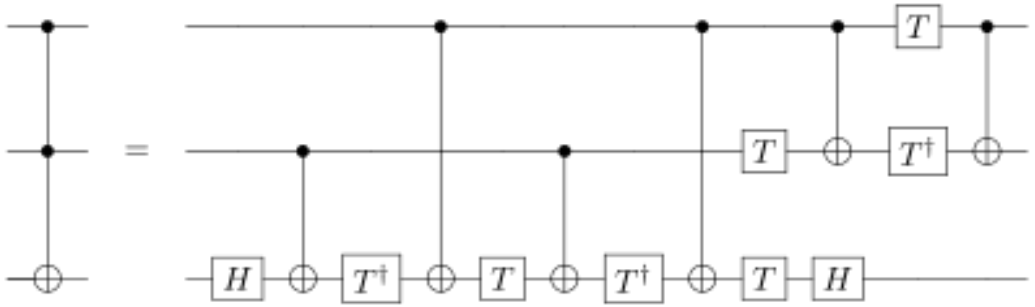
Table 2: Polynomials considered for RQC. P is the polynomial used to define $\mathbb{F}_{q^m}^n$ as $\mathbb{F}_{q^m}[X]/\langle P\rangle$ and Π is the polynomial used to define $\mathbb{F}_{q^m}$ as $\mathbb{F}_q[X]/\langle\Pi\rangle$.

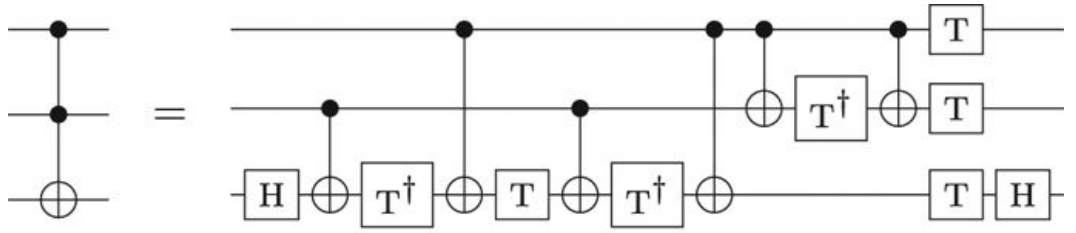
Toffoli

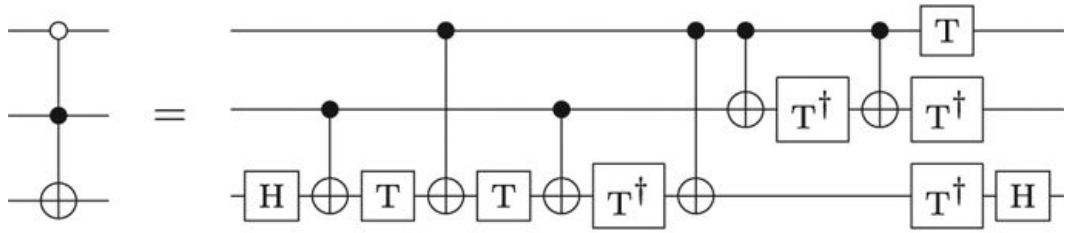
$$Toffoli = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

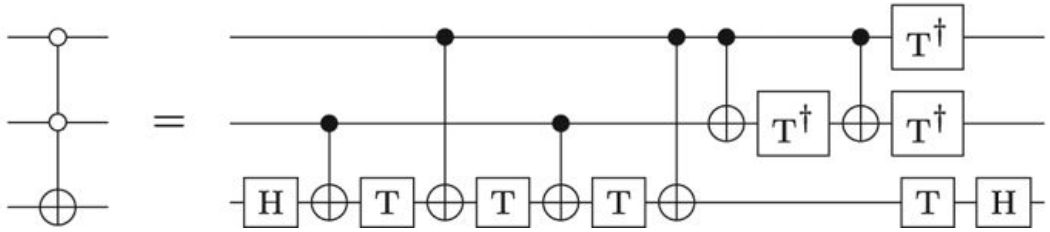


Toffoli



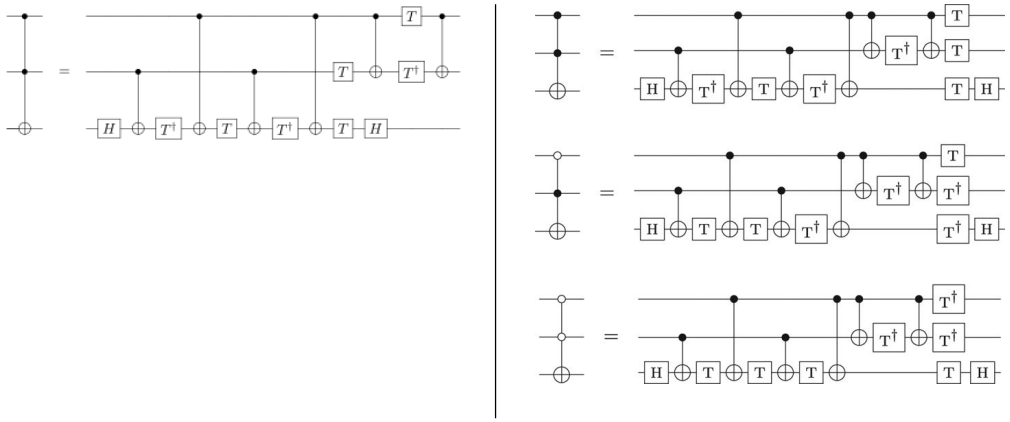






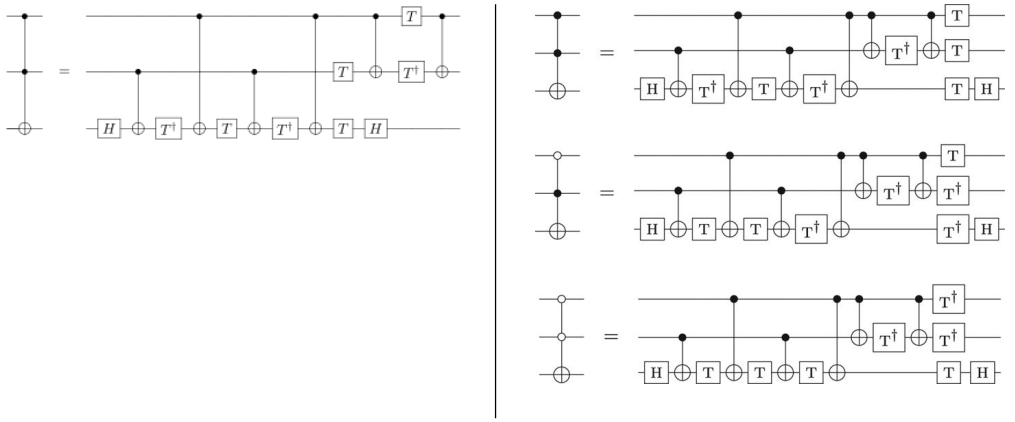
https://www.nature.com/articles/s41534-018-0072-4#Sec8

Toffoli



- Toffoli gate 내부의 연산 게이트 수를 줄일 수 있는 것이 가능한가

Toffoli



exp() 함수는 Euler 의 상수 e (약 2.71828) 값을 입력 받은 인자 값만큼 거듭제곱 하는 함수 입니다.
즉, 입력 인자 값은 double 형이며 이 인자 값이 e 의 승수를 나타냅니다.

Syntax

public static double exp (double a)

Example

```
1 System.out.println(Math.exp(2)); cs
```

위 코드는 2.71828... 의 2승을 한 값을 리턴 합니다.
결과 값은 7.38905609893065 입니다.

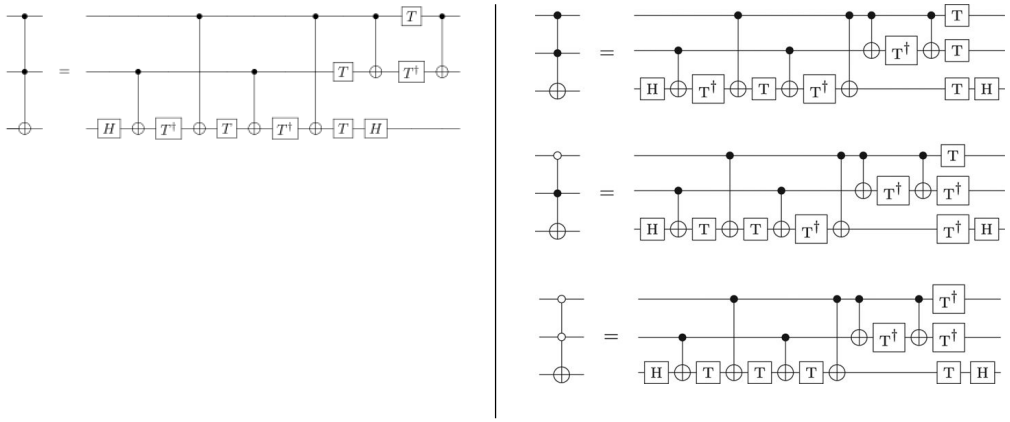
$$\begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{i\pi}{4}\right) \end{pmatrix}$$

- Toffoli gate 내부의 연산 게이트 수를 줄일 수 있는 것이 가능한가

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{-i\pi}{4}\right) \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{i\pi}{4}\right) \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{-i\pi}{4}\right) \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{-i\pi}{4}\right) \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{i\pi}{4}\right) \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{i\pi}{4}\right) \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Toffoli



- Toffoli gate 내부의 연산 게이트 수를 줄일 수 있는 것이 가능한가
- Karatsuba 등등

감사합니다