JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 24, No. 8: 1044~1051, Aug. 2020

부호 기반 양자 내성 암호의 이진 필드 상에서 곱셈 연산 양자 게이트 구현

최승주[1] · 장경배[1] · 권혁동[1] · 서화정[2*]

Implementation of Quantum Gates for Binary Field Multiplication of Code based Post Quantum Cryptography

Seung-Joo Choi[1] · Kyong-Bae Jang[1] · Hyuk-Dong Kwon[1] · Hwa-Jeong Seo[2*]

[1]Graduate Student, Department of IT Fusion Engineering, Hansung University, Seoul, 02876 Korea
[2*]Associate Professor, Department of IT Fusion Engineering, Hansung University, Seoul, 02876 Korea

요 약

양자 컴퓨터의 시대가 점점 현실로 다가오고 있다. 이에 대비해 미국 국립 표준 기술 연구소에서는 양자 알고리즘으로부터 내성이 있는 양자 내성 암호의 표준을 정하기 위해 후보군을 모집했다. 제출된 암호들은 양자 알고리즘으로부터 안전할 것으로 예상이 되지만 알고리즘이 실제 양자 컴퓨터상에서 작동이 되었을 때에도 양자 알고리즘의 공격으로부터 안전한지 검증을 할 필요가 있다. 이에 본 논문에서는 부호 기반 양자 내성 암호의 이진 필드 상에서의 곱셈 연산을 양자 컴퓨터에서 작동될 수 있게 양자 회로로 구현하였고 해당 회로를 최적화 하는 방안에 대하여 설명한다. 구현은 대표적인 부호 기반 암호인 Classic McEliece에서 제시하는 2개의 필드 다항식과 ROLLO에서 제시하는 3개의 필드 다항식에 대하여 일반 곱셈 알고리즘과 카라추바 곱셈 알고리즘으로 구현하였다.

ABSTRACT

The age of quantum computers is coming soon. In order to prepare for the upcoming future, the National Institute of Standards and Technology has recruited candidates to set standards for post quantum cryptography to establish a future cryptography standard. The submitted ciphers are expected to be safe from quantum algorithm attacks, but it is necessary to verify that the submitted algorithm is safe from quantum attacks using quantum algorithm even when it is actually operated on a quantum computer. Therefore, in this paper, we investigate an efficient quantum gate implementation for binary field multiplication of code based post quantum cryptography to work on quantum computers. We implemented the binary field multiplication for two field polynomials presented by Classic McEliece and three field polynomials presented by ROLLO in generic algorithm and Karatsuba algorithm.

키워드 : 양자 게이트, 이진 필드 곱셈, 부호 기반 양자 내성 암호, IBM ProjectQ

Keywords : Quantum Gates, Binary Field Multiplication, Code based Post Quantum Cryptography, IBM Project Q

Received 2 April 2020, **Revised** 9 April 2020, **Accepted** 20 May 2020

*** Corresponding Author** Hwajeong Seo (E-mail:hwajeong84@gmail.com, Tel: +82-2-760-8033)
Associate Professor, Department of IT Fusion Engineering, Hansung University, Seoul, 02876 Korea

Open Access http://doi.org/10.6109/jkiice.2020.24.8.1044 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

양자 컴퓨터의 시대가 점점 다가오고 있다. 양자 컴퓨터는 1980년대 초반 미국의 물리학자 R. Feynman이 아이디어를 제안하면서 등장하였고[1] 40년간 꾸준한 연구를 통해 실제 양자 컴퓨터가 2019년 IBM에 의해 개발이 되었다[2]. 이와 같은 양자의 중첩 상태를 이용하여 연산을 진행하는 해당 컴퓨터의 특징을 이용하면 과거 수년간의 연산 시간이 걸리던 문제들을 단 몇 초안에 풀 수 있을 것이라고 예상이 되고 있다. 그러나 이러한 양자 컴퓨터의 연산 능력은 현존하는 공개키 암호인 RSA, ECC 등에도 위협이 되고 있다. 이에 대비하기 위하여 미국 국립 표준 기술 연구소(NIST; National Institute of Standards and Technology)에서는 양자 내성 암호 표준 후보군을 선출하기 위한 공모전을 진행하고 있다. 후보군으로는 격자, 다변수 다항식, 타원 곡선, 해시 기반 암호 등 다양한 암호들이 제출되었는데 이들 중 가장 오랫동안 보안상으로 치명적인 취약점이 없었던 부호 기반 암호 또한 제출되었다. 제출된 암호들은 NIST에서 요구하는 양자 컴퓨터로부터도 안전한 암호 알고리즘이 되기 위한 조건들에 부합되기 위해 계속해서 개선되고 있다. 그러나 해당 조건을 만족시킨다 하더라도 실제 양자 컴퓨터에서 양자 알고리즘을 이용하여 해당 암호들을 공격 하였을 때에도 안전할 것이라 보장할 수는 없다. 현재 구현되어 있는 양자 내성 암호들은 전부 기존 컴퓨터에서 작동될 수 있게 C 언어 등으로 구현이 되어있는 상태이다. 이와 같은 양자 내성 암호들에 대한 제대로 된 보안 강도를 알기 위해서는 실제 양자 컴퓨터상에서 작동될 수 있게 구현하여 안전성을 확인할 필요가 있다. 이를 위해서 본 논문에서는 가장 오랫동안 보안성이 보장되었던 부호 기반 암호 중 하나인 Classic McEliece의 이진 필드 곱셈 연산을 IBM ProjectQ를 이용하여 양자 컴퓨터에서 동작될 수 있게 양자 회로로 구현하였으며 또 다른 부호 기반 양자 내성 암호 후보인 ROLLO의 곱셈 연산 또한 구현하였다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 양자 게이트에 대한 설명과 이진 필드 곱셈 연산에 대한 이전 연구들에 대한 간략한 소개를 하며 구현 환경인 IBM ProjectQ에 대한 설명을 한다. 3장에서는 Classic McEliece와 ROLLO의 이진 필드 곱셈 대한 일반 곱셈 방식과 카라추바 방식을 이용한 구현 방법에 대한 설명을 한다. 4장에서는 해당 구현 결과에 대한 분석을 진행한다. 마지막 5장에서는 본 논문의 결론을 맺는다.

Ⅱ. 관련 연구

2.1. 이진 필드 곱셈 양자 게이트

양자 컴퓨터 또한 기존 컴퓨터에 존재하는 게이트들을 표현할 수 있는 여러 개의 게이트가 존재한다[3]. 본 논문에서 구현한 이진 필드 곱셈 연산에서 사용되는 양자 게이트들은 표 1과 같다.

Table. 1 Quantum gates for binary field multiplicatoin

Gates	Operation	Notation
Toffoli	And or F_2 multiplication	(a, b, c) -> (a, b, c⊕(a·b))
CNOT	XOR	(a, b) -> (a, a⊕b)

2.2. 이진 필드 곱셈 알고리즘

이진 필드 곱셈에 대한 효율적이면서 일반적인 연산 방법은 [4]에 나와 있다. 해당 논문의 연산 방식은 먼저 일반 적인 곱셈 연산을 통해 모듈러 연산을 취해주어야 하는 차수가 높은 값들을 먼저 도출해낸다. 그 후 해당 값들에 대한 모듈러 연산을 취해주고 차수가 낮아 모듈러 연산이 필요 없는 부분들에 대한 곱셈 연산을 진행하는 방식을 사용한다. 이러한 방식을 통해 빠른 이진 필드 곱셈 연산이 가능하다.

이진 필드 곱셈에 대한 또 다른 연산 방식으로는 카라추바[5] 연산 방식이 있다. 일반적으로 두 n자리 수의 곱셈은 n^2에 비례하는 횟수의 연산이 필요한데 카라추바 알고리즘을 사용하면 이를 $n^{log2^{\wedge}3}$ 회로 줄이는 것이 가능하다. 해당 알고리즘을 양자 내성 암호의 이진 필드 곱셈 방식에 적용을 하여 연산 최적화가 가능하다. 그러나 카라추바를 적용하기 위해서는 일반적인 곱셈 방식에 비해 추가적으로 발생되는 중간 결과 값들을 저장하기 위해 큐비트를 더 할당해야 한다는 단점이 있다.

2.3. IBM ProjectQ

ProjectQ는 IBM에서 제공하는 양자 컴퓨팅을 위한 오픈 소스 소프트웨어 프레임워크이다[6]. 해당 프레임워크를 이용하여 IBM에서 제작한 양자 칩에서 양자 프

로그램을 작동시켜 볼 수 있으며 기존 컴퓨터에서 양자 프로그래밍을 시뮬레이션 할 수도 있게 지원한다. 해당 프레임워크는 발전 중에 있는 상태에 있기 때문에 작동시킬 수 있는 큐비트의 개수에 한계가 있다.

본 논문에서는 Classic McEliece[7]와 ROLLO[8]가 제시하는 필드 다항식에 대한 이진 필드 곱셈에 대하여 일반 곱셈 방식과 카라추바 방식에 대한 구현 및 평가한다.

Ⅲ. 구 현

이진 필드 곱셈에 대한 구현은 Classic McEliece에서 제시하는 필드 다항식 ($x^{12} + x^3 + 1$)과 ($x^{13} + x^4 + x^3 + x + 1$) 그리고 ROLLO에서 제시하는 필드 다항식 ($x^{47} + x^5 + 1$), ($x^{53} + x^6 + x^2 + x + 1$) 그리고 ($x^{67} + x^5 + x^2 + x + 1$)에 대하여 일반 곱셈 방식과 카라추바 방식으로 진행하였다. 구현에 대한 구체적인 설명은 Classic McEliece의 필드 다항식 ($x^{12} + x^3 + 1$)에 대하여만 설명한다.

3.1. 일반 이진 필드 곱셈

($x^{12} + x^3 + 1$)을 일반 이진 필드 곱셈 방식으로 연산하기 위해서는 총 36개의 큐비트가 필요하다. 이 중 24개는 피연산자 다항식을 나타내며($a_0 \sim a_{11}$, $b_0 \sim b_{11}$) 나머지 12개는 연산 중 발생되는 중간 값과 최종 결과 값을 저장하는 공간($c_0 \sim c_{11}$)으로 사용된다.

먼저 두 다항식간 곱셈 연산을 진행했을 때 모듈러 연산이 필요한, 즉 곱했을 때 차수가 12 이상이 되는 값들($a_6 \sim a_{11}$, $b_6 \sim b_{11}$)에 대하여 곱셈 연산을 진행하며 이에 대한 연산 과정은 알고리즘 1과 같다. 곱셈은 두 다항식을 곱했을 때 차수가 12가 되는 연산자인 (a_{11}, b_1), (a_{10}, b_2),...,(a_2, b_{10}), (a_1, b_{11})의 값을 곱하여 c_0부터 넣고 그 다음 c_1에는 차수가 13이 되는 연산자들의 곱을 넣는 형식으로 진행이 된다. 이와 같은 곱셈 연산은 두 다항식의 곱으로 나올 수 있는 최대 차수 22, 즉 (a_{11}, b_{11})의 곱셈 결과가 나올 때까지 진행된다. 해당 연산이 완료가 되면 각각의 중간 값은 ($c_0 \sim c_{10}$)에 들어가게 된다.

곱셈이 완료가 되면 해당 결과 값들에 대하여 모듈러 연산을 진행하며 모듈러 연산 과정은 표 2와 같다. 표 2에서 첫 번째 행은 방금 연산한 두 다항식간의 곱의 결과로 나오는 차수를 나타낸다. 앞서 언급한 곱셈의 결과로 나올 수 있는 최소 차수 x^{12}부터 최대 차수 x^{22}까지 있는 것을 확인할 수 있다.

Algorithm. 1 Generic binary multiplication for higher part on x12 + x3 + 1

Input: Two operand polynomials: a[12], b[12]
Output: Result values of multiplication: c[12]

```
1:  t ← 12          //t: Target polynomial coefficient
2:  c ← t - 1
3:  s ← t - 2
4:  d ← s
5:  k ← 0
6:  For i = 0  to c do
7:     For z = c, z > k, z decrease do
8:        c[i] ← c[i] ⊕ (a[z]·b[c-d])
9:        d ← d-1
10: k ← k+1
11: s ← s-1
12: d ← s
```

모듈러 연산 과정을 진행하면서 연산 횟수에 대한 최적화를 할 수 있다. 연산 과정에서 x^0을 연산하기 위해 사용되는 x^{12}과 x^{21}에 해당하는 값이 x^3에서도 사용되는 것을 표 2의 2번째 행과 5번째 행에서 볼 수 있다. 이와 같은 현상은 x^1과 x^4 사이에서도 볼 수 있다(x^{13}, x^{22}). 이를 이용해 본 논문에서는 먼저 x^0과 x^1에 대한 연산을 진행하고 해당 결과 값을 x^3과 x^4를 연산할 때 사용하여 x^3과 x^4를 연산하기 위해 각각 3번의 연산이 필요한 부분을 2번으로 줄였다.

모듈러 연산이 완료가 되면 곱의 차수가 11 이하가 되는 값들($a_0 \sim a_5$, $b_0 \sim b_5$)에 대하여 곱셈 연산을 진행하며 과정은 알고리즘 2와 같다. 곱셈 후 해당 결과 값을 ($c_0 \sim c_{10}$)에 넣어 일반 곱셈 연산을 완료하며 구현한 회로는 그림 1과 같다.

이와 같은 일반 곱셈 방식의 모듈러 연산에서 ($x^{12} + x^3 + 1$)나 ($x^{47} + x^5 + 1$)상에서는 모듈러 연산 결과를 담을 큐비트를 제외하고 추가적인 큐비트가 필요하지 않다. 그러나 ($x^{13} + x^4 + x^3 + x + 1$), ($x^{53} + x^6 + x^2 + x + 1$) 그리고 ($x^{67} + x^5 + x^2 + x + 1$)을 연산함에 있어 다항식의 항이 많아 추가적인 큐비트가 필요하다. 하여 본 논문의 4장에 나오는 표 3에 나오는 일반 곱셈 연산 시 필요한 큐비트의 개수를 확인하면 x^{12}와 x^{47}은 각각 정확히 필드 다항식 차수의 3배만큼의 큐비트가 사용되는 것을 확인할 수 있다. 그에 반해 x^{13}, x^{53}와 x^{67}과 같은 경우 필드 다

항식 차수의 3배 만큼에 추가적인 큐비트가 더 필요한 것을 볼 수가 있다.

Table. 2 Modular reduction for binary multiplication on $x^{12} + x^3 + 1$

	x^{12}	x^{13}	x^{14}	x^{15}	x^{16}	x^{17}	x^{18}	x^{19}	x^{20}	x^{21}	x^{22}
1	c_0									c_9	
x^1		c_1									c_{10}
x^2			c_2								
x^3	c_0			c_3						c_0	
x^4		c_1			c_4						c_1
x^5			c_2			c_5					
x^6				c_3			c_6				
x^7					c_4			c_7			
x^8						c_5			c_8		
x^9							c_6			c_9	
x^{10}								c_7			c_{10}
x^{11}									c_8		

Algorithm. 2 Generic binary multiplication for lower part on $x^{12} + x^3 + 1$

Input: Two operand polynomials: a[12], b[12]
Output: Result values of multiplication: c[12]

```
1: For i = 0 to 12 do
2:     For z = i, z ≥ 0, z decrease do
3:         c[i] ← c[i] ⊕ (a[z]·b[i-z])
```

3.2. 카라추바 이진 필드 곱셈

카라추바 곱셈 방식 같은 경우 총 47개의 큐비트가 필요하다. 이 중 24개는 피연산자 다항식을 나타내며($a_0 \sim a_{11}$, $b_0 \sim b_{11}$) 나머지 12개는 연산 중 발생되는 중간 값과 최종 결과 값을 저장하는 공간($c_0 \sim c_{22}$)으로 사용된다. 카라추바 방식은 일반 곱셈 방식과는 다르게 두 다항식에 대한 모든 곱셈을 완료한 후에 모듈러 연산을 진행한다. 해당 곱셈의 연산은 다음과 같이 진행된다.

먼저 크기가 n인 두 다항식 f와 g 그리고 카라추바 곱셈의 결과 값 h에 대해 $f = f_0 + f_1x^k$, $g = g_0 + g_1x^k$, $h = h_0 + h_1x^k + h_2x^{2k} + h_3x^{3k}$($n/2 \leq k < n$)로 나누어 표현한다. 그 후 결과 값 h를 구하기 위해 A, B, R 값을 연산한다($A = f_0 \cdot g_0$, $B = f_1 \cdot g_1$, $R = (f_0 + f_1)\cdot(g_0 + g_1)$). 마지막으로 연산된 A, B, R 값을 이용해 $h + f \cdot g = h + A + (R + A + B)x^k + Bx^{2k}$를 연산하여 카라추바 곱을 수행한다. 이때 두 다항식의 낮은 차수 부분과 높은 차수 부분들의 곱인 $A(f_0 \cdot g_0)$,와 $B(f_1 \cdot g_1)$과 $R((f_0 + f_1)\cdot(g_0 + g_1))$에 대해서 곱의 연산으로 인해 낮은 차수들의 곱셈의 결과가 높은 차수 곱의 결과와 겹치는 현상이 발생할 수 있다. 하여 해당 A와 B그리고 R도 카라추바 방식을 이용해 나눠 연산을 한다. 그럼 최종적으로 계산되어야 하는 식은 $h + f \cdot g = (h_0 + A_0) + (h_1 + A_0 + A_1 + B_0 + R_0)x^k + (h_2 + A_1 + B_0 + B_1 + R_1)x^{2k} + (h_3 + B_1)x^{3k}$이 된다.

카라추바 곱셈 연산의 순서는 표 3과 같으며 필드 다항식 ($x^{12} + x^3 + 1$)을 기준으로 A_0, B_0, R_0는 6자리의 수, A_1, B_1 그리고 R_1은 5자리의 수로 나누어 연산을 진행한다.

Table. 3 Karatsuba multiplication order on $x^{12} + x^3 + 1$

	h_0+A_0	$h_1+A_0+A_1+B_0+R_0$	$h_2+A_1+B_0+B_1+R_1$	h_3+B_1
1	A_0			
2		A_1		
3		A_1+B_0		
4				B_1
5			A_1+B_0	
6		$A_0+A_1+B_0$		
7			$A_1+B_0+B_1$	
8		$A_0+A_1+B_0+R_0$		
9			$A_1+B_0+B_1+R_1$	

A_0와 A_1과 같은 경우 두 피연산자 다항식의 차수가 낮은 절반($a_0 \sim a_5$, $b_0 \sim b_5$)을 사용해 값을 구한다. A_0는 알고리즘 2, A_1는 알고리즘 3의 연산을 통해 A_0에 이어 값을 구한다. 해당 값들은 ($c_0 \sim c_5$)에 저장이 된다.

Algorithm. 3 Karatsuba binary multiplication for creating A_1 on $x^{12} + x^3 + 1$

Input: Lower part of polynomials: a[0-5], b[0-5]
Output: Result values of multiplication: c[0-5]

```
 1: t ← 6
 2: a ← 6
 3: c ← t - 1
 4: s ← t - 2
 5: d ← s
 6: k ← 0
 7: For i = 0 to c do
 8:     For z = c, z > k, z decrease do
 9:         c[a] ← c[a] ⊕ (a[z]·b[c-d])
10:         d ← d-1
11: a ← a+1
12: k ← k+1
13: s ← s-1
14: d ← s
```

B_0와 B_1과 같은 경우 두 피연산자 다항식의 차수가 높은 절반(a_6~a_{11}, b_6~b_{11})을 사용해 값을 구한다. B_0는 알고리즘 2와 동일한 연산을 이용하되 해당 결과 값을 A_1이 저장되어 있는 (c_6~c_{11})에 담는다. 그 이유는 표 3을 보면 A_1과 B_0는 항상 같이 연산이 되어 사용이 되기 때문이다. 이런 특징을 활용하여 표 3의 단계 5를 보면 단계 3에서 형성한 (A_1+B_0)의 값을 재활용 하여 연산 횟수를 줄이는 것을 볼 수 있다. B_1는 같은 경우 알고리즘 4의 연산의 통해 B_0에 이어 값을 구하며 해당 결과 값은 (c_{18}~c_{22})에 담는다.

A_0, A_1, B_0, B_1의 모든 값을 구했으면 표 3의 단계 5-7과 같이 할당되어야 하는 위치에 알맞게 값들을 넣어준다.

곱셈 연산의 마지막으로 R의 값을 구해 할당되어야 하는 자리에 넣어준다. 이때 R을 구하기 위한 연산인 (f_0 + f_1)와 (g_0 + g_1)를 해 주는데 피연산자 다항식 낮은 절반(a_0~a_5, b_0~b_5) 부분과 높은 절반(a_6~a_{11}, b_6~b_{11})을 CNOT 게이트를 통해 XOR 연산을 취한다. R을 구한 후에 동일한 연산을 다시 수행해 알고리즘의 역이 가능하도록[9] 원래 상태로 되돌려 놓는다.

카라추바 곱셈 연산이 완료가 되면 필드 다항식에 맞춰 차수에 알맞은 모듈러 연산을 알고리즘 4를 통해 진행한다. 카라추바 곱셈을 구현한 회로는 그림 2와 같다.

Algorithm. 4 Karatsuba multiplication modular reduction on $x^{12} + x^3 + 1$

CNOT(A, B): B ← A ⊕ B

1:	CNOT(c[12], c[0])	13:	CNOT(c[18], c[6])
2:	CNOT(c[12], c[3])	14:	CNOT(c[18], c[9])
3:	CNOT(c[13], c[1])	15:	CNOT(c[19], c[7])
4:	CNOT(c[13], c[4])	16:	CNOT(c[19], c[10])
5:	CNOT(c[14], c[2])	17:	CNOT(c[20], c[8])
6:	CNOT(c[14], c[5])	18:	CNOT(c[20], c[0])
7:	CNOT(c[15], c[3])	19:	CNOT(c[20], c[3])
8:	CNOT(c[15], c[6])	20:	CNOT(c[21], c[9])
9:	CNOT(c[16], c[4])	21:	CNOT(c[21], c[1])
10:	CNOT(c[16], c[7])	22:	CNOT(c[21], c[4])
11:	CNOT(c[17], c[5])	23:	CNOT(c[22], c[10])
12:	CNOT(c[17], c[8])	24:	CNOT(c[22], c[2])
		25:	CNOT(c[22], c[5])

Classic McEliece의 필드 다항식 ($x^{12} + x^3 + 1$)과 ($x^{13} + x^4 + x^3 + x + 1$) 및 ROLLO의 필드 다항식 ($x^{47} + x^5 + 1$), ($x^{53} + x^6 + x^2 + x + 1$) 그리고 ($x^{67} + x^5 + x^2 + x + 1$)에 대하여도 앞서 설명한 두 가지 곱셈 방식 모두를 적용하여 구현하였다.

Ⅳ. 평 가

이진 필드 곱셈에 대한 구현은 IBM ProjectQ를 이용하여 구현하였다. 구현 언어는 파이썬 3.6을 사용하였고 인텔 NUC 10 미니 PC 환경에서 Ubuntu 16.04 운영체제를 사용했다. 2장에서 언급하였던 것과 같이 현재 IBM ProjectQ는 개선단계에 있는 중이기에 연산할 수 있는 큐비트의 수에 한계가 있다. 그렇기 때문에 IBM ProjectQ에서 연산할 수 있는 필드 다항식 값들에 대해서는 ProjectQ에서 연산을 진행하였고 큐비트 수가 많아 작동될 수 없는 다항식 값들에 대해서는 동일한 연산을 C 언어로 구현하여 이진 필드 곱셈 연산 결과 값을 확인하고 사용되는 게이트 자원의 수를 측정했다. 해당 결과는 표 4에 정리가 되어있고 CNOT 게이트와 토폴리(Toffoli) 게이트 각각의 사용량에 대한 그래프는 그림 3과 그림 4에 나와 있다.

표 4에서 볼 수 있듯이 카라추바 곱셈 방식을 사용할 경우 일반 곱셈 방식보다 큐비트가 더 많이 필요한 것을 볼 수 있다. 그러나 토폴리 게이트의 수는 필드 다항식의 차수가 크면 클수록 일반 곱셈 방식에서 사용하는 토폴리 게이트 수가 카라추바 방식에서 사용하는 게이트의 수 보다 지속적으로 더 많아지는 것을 그림 4를 보면 알 수 있다. 특히 토폴리 게이트 하나가 6개의 CNOT 게이트와 9개의 1큐비트 게이트(i.e Hadamard, T)로 이루어져 있다는 점을 감안하면[10] 카라추바 곱셈 방법이 게이트 수에 있어서 일반 곱셈 방식보다 더 효율적인 것을 알 수 있으며 이러한 현상은 필드 다항식의 차수가 높을수록 효율성의 차이를 더 명확히 볼 수 있다.

Table. 4 Resource count result

	Generic			Karatsuba		
Resource	Qubit	Toffoli	CNOT	Qubit	Toffoli	CNOT
x^{12}	36	144	11	47	108	66
x^{13}	43	169	41	51	134	97
x^{47}	141	2,209	46	187	1,681	257
x^{53}	164	2,809	160	211	2,134	406
x^{67}	205	4,489	201	267	3,401	508

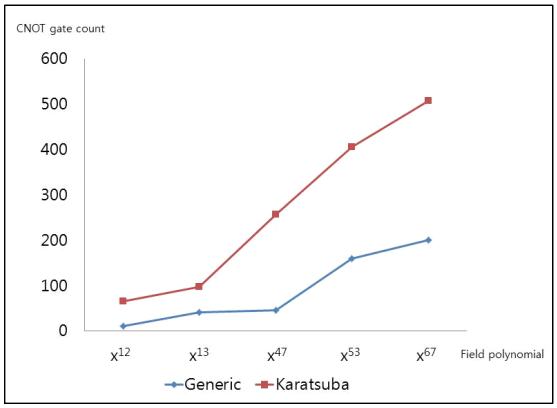

Fig. 3 Number of CNOT gate used during binary multiplication

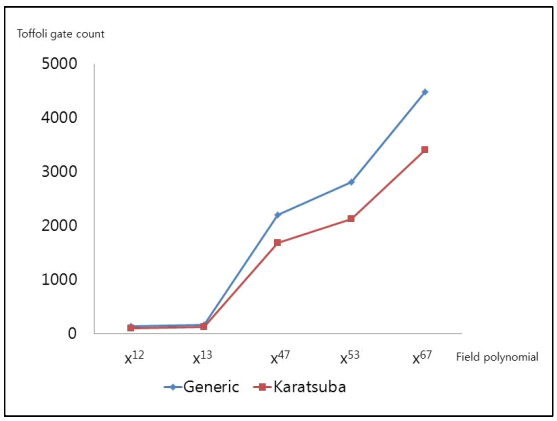

Fig. 4 Number of Toffoli gate used during binary multiplication

Ⅴ. 결 론

양자 컴퓨터의 시대가 가까워짐에 따라 이에 대한 암호학적 대비 또한 동시에 준비되고 있다. 이에 본 논문에서는 실제 양자 컴퓨터상에서 작동될 수 있게 양자 내성 암호 중 부호 기반 암호들의 이진 필드 곱셈 연산을 일반 곱셈 방식과 카라추바 곱셈 방식으로 IBM PorjectQ에서 양자 회로로 구현하여 각 방식의 자원 사용량에 대해 비교 분석하였다. 또한 일반 곱셈 연산의 모듈러 연산 횟수를 최적화 하였다.

향후 연구로는 카라추바 알고리즘의 큐비트 사용량을 줄일 수 있는 방안을 연구할 예정이며 전체적인 이진 필드 곱셈 연산에서 사용되는 게이트 수를 줄일 수 있는 방안을 연구를 할 예정이다.

ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (<Q|Crypton>, No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity)

References

[1] R. Feynman, “Simulating Physics with Computers,” *International Journal of Theoretical Physics*, 21, 467, 1982.

[2] Join Extra Crunch. IBM unveils its first commercial quantum computer [Internet]. Available: https://techcrunch.com/2019/01/08/ibm-unveils-its-first-commercial-quantum-computer/.

[3] A. G. Aruna, K. H. Vani, C. Sathya, and R. Sowndarya Meena, “A Study on Reversible Logic Gates of Quantum Computing,” *International Journal of Computer Science and Information Technologies*, vol. 7, no. 1, pp. 427-432. 2016.

[4] D. Cheung, D. Maslov, J. Mathew, and D. K. Pradhan, “On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography,” *In Workshop on Quantum Computation, Communication, and Cryptography*, pp. 96-104, 2008.

[5] S. Kepley, and R. Steinwandt, “Quantum circuits for multiplication with subquadratic gate count,” *Quantum Information Processing*, vol. 14, no. 7, pp. 2373-2386, 2015.

[6] D. S. Steiger, T. Haner, and M. Troyer, “ProjectQ: An Open Source Software Framework for Quantum Computing. Quantum,” *ResearchGate*, 2. 10.22331/q-2018-01-31-49, 2016.

[7] D. J. Bernstein, T. Chou, T. Lange, I. V. Maurich, R. Misoczki, R. Niederhagen, and J. Szefer, “Classic McEliece: conservative code-based cryptography,” *NIST submissions*, 2017.

[8] C. A. Melchor, N. Aragon, and M. Bardet, “ROLLO: Rank-Ouroboros, Lake & Locker,” *NIST submissions*, 2019.

[9] I. V. Hoof, “Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count,” *arXiv preprint arXiv*:1910.02849, 2019.

[10] V. Shende, and I. L. Markov, “On the CNOT-cost of TOFFOLI gates,” *Quantum Information and Computation*, 2008.

APPENDIX

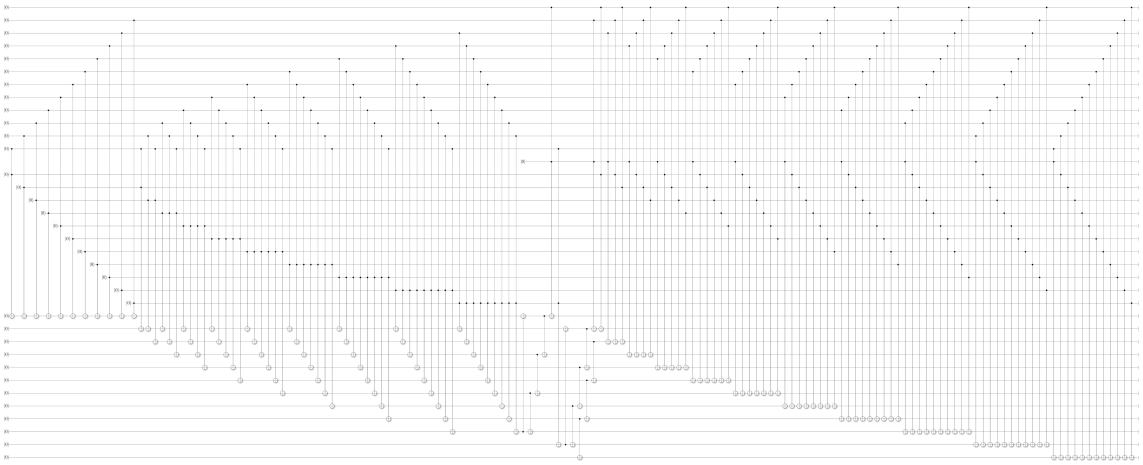

Fig. 1 Circuit for GF(2^{12}) multiplier with $P(x) = x^{12} + x^3 + 1$ in general multiplication

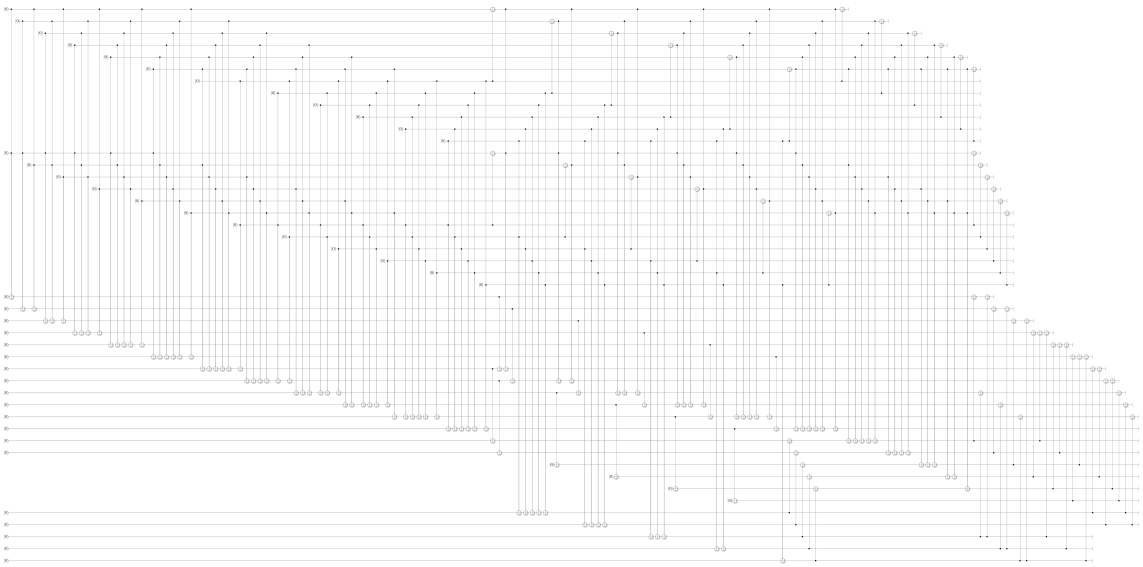

Fig. 2 Circuit for GF(2^{12}) multiplier with $P(x) = x^{12} + x^3 + 1$ in Karatsuba multiplication

최승주(Seung-Joo Choi)

2019년 2월: 한성대학교 영어영문학과 학사
2019년 3월~현재: 한성대학교 IT융합공학과 석사과정
※관심분야 : 블록체인, IoT

장경배(Kyeong-Bae Jang)

2019년 2월: 한성대학교 IT응용시스템공학과 공학 학사 졸업
2019년 8월~현재: 한성대학교 IT융합공학과 석사과정
※관심분야 : IoT, 정보보안

권혁동(Hyuk-Dong Kwon)

2018년 2월: 한성대학교 정보시스템공학과 공학 학사
2020년 2월: 한성대학교 IT융합공학과 석사 졸업
2020년 3월~현재: 한성대학교 정보컴퓨터공학과 박사과정
※관심분야 : 블록체인, 암호구현

서화정(Hwa-Jeong Seo)

2010년 2월 부산대학교 컴퓨터공학과 학사 졸업
2012년 2월 부산대학교 컴퓨터공학과 석사 졸업
2012년 3월~2016년 1월: 부산대학교 컴퓨터공학과 박사 졸업
2016년 1월~2017년 3월: 싱가포르 과학기술청
2017년 4월~현재: 한성대학교 IT 융합공학부 조교수
※관심분야 : 정보보호, 암호화 구현, IoT