HQC PKE의 핵심 연산에 대한 양자회로 최적 구현

임세진[1], 장경배[2], 오유진[3], 서화정[4]
[1]한성대학교 IT융합공학과 석사과정
[2]한성대학교 정보컴퓨터공학과 박사과정
[3]한성대학교 융합보안학과 석사과정
[4]한성대학교 융합보안학과 교수
dlatpwls834@gmail.com, starj1023@gmail.com, oyj0922@gmail.com,
hwajeong84@gmail.com

Optimal implementation of quantum circuits for HQC's PKE core operations

Se-Jin Lim[1], Kyung-Bae Jang[2], Yu-Jin Oh[3], Hwa-Jeong Seo[4]
[1]Dept. of IT Convergence Engineering, Han-Sung University
[2]Dept. of Information and Computer Engineering, Han-Sung University
[3]Dept. of Convergence Security, Han-Sung University
[4]Dept. of Convergence Security, Han-Sung University

요 약

양자 컴퓨터가 빠르게 발전됨에 따라 기존의 공개키 암호들이 기반하고 있는 난제인 소인수분해, 이산로그 문제를 다항 시간 안에 풀 수 있는 Shor 알고리즘에 의해 기존 암호의 보안 강도 약화 및 무력화 시기가 다가오고 있다. NIST에서는 양자 컴퓨터 시대에 대비하여 양자 컴퓨터가 등장하더라도 안전한 암호인 양자내성암호에 관한 공모전을 개최하였다. 양자 컴퓨터 환경에서 암호 분석을 통해 암호의 보안 강도를 확인할 수 있는데, 이를 위해서는 암호를 양자회로로 구현해야한다. 본 논문에서는 NIST PQC 공모전의 4 라운드 후보 알고리즘인 HQC (Hamming Quasi-Cyclic)의 PKE (Public Key Encryption) 버전에 대한 키 생성 및 인코딩 연산 중 핵심 역할을 하는 바이너리 필드 산술과 shortened Reed-Solomon 코드의 인코딩 연산에 대한 최적화된 양자회로 구현을 제안하고, 이를 위해 필요한 자원을 추정한다.

1. 서론

중첩과 얽힘 성질에 의해 매우 빠른 연산 속도를 가지는 양자 컴퓨터는 머신러닝, 금융, 최적화와 같은 분야에서 큰 이점을 가져올 것으로 기대되어 매우 활발히 개발되고 있다. 반면 현재 사용되는 공개키 암호 측면에서는 양자 컴퓨터상에서 동작하는 Shor 알고리즘[1]에 의해 공개키 암호가 기반하고 있는 소인수분해, 이산로그 문제와 같은 수학적 난제를 다항 시간 안에 해결할 수 있으므로 보안성이 붕괴되는 시기가 다가오고 있다고 볼 수 있다. 따라서 NIST에서는 양자 컴퓨터 시대에 대비하기 위해 양자내성암호 공모전을 개최하여 양자 컴퓨터가 등장하더라도 보안성을 유지할 수 있는 새로운 공개키 암호 표준화를 진행하고 있다. 양자 컴퓨터 환경에서 암호 분석을 통해 암호의 보안강도를 확인할 수 있는데, 이를 위해서는 암호를 양자회로로 구현해야한다. 본 논문에서는 해당 공모전의 4 라운드 후보 알고리즘인 HQC (Hamming Quasi-Cyclic)[2]의 PKE (Public Key Encryption) 버전에 대한 키 생성 및 인코딩 연산 중 핵심 역할을 하는 바이너리 필드 산술과 shortened Reed-Solomon[3] 코드의 인코딩 연산에 대한 최적화된 양자회로 구현을 제안하고, 이를 위해 필요한 자원을 추정한다.

2. 관련 연구

2.1 코드 기반 암호

양자내성암호는 기반으로 하는 난제에 따라 여러 가지로 분류할 수 있는데, 본 논문에서 대상으로 삼은 암호인 HQC[2]는 코드 기반 암호에 속한다. 코드 기반 암호는 메시지에 의도적으로 오류를 주입하고 오류를 알고 있는 사용자만 오류 정정 코드를 사용하여 메시지를 복원할 수 있게 한다는 원리이며, NP-complete에 해당하는 난제인 신드롬 디코딩 문제를 기반으로 두고 있다. 신드롬 디코딩은 오류가 있는 코드 워드에서 발생하는 신드롬 정보를 활용하여 오류를 수정하는 작업이다. 신드롬 디코딩 문제는

$s^\top = He^\top$ (s와 H, e는 모두 바이너리 필드에 속한다.)로 표현되며, 신드롬 값 s는 데이터와 오류 정정 비트로 구성된 블록인 코드 워드 e와 오류 정정 코드의 패리티 체크 행렬 H의 곱을 통해 생성된다[2]. 이때 H와 $s^\top$를 알고 있다고 해도 e를 알아내는 것은 굉장히 어려운 작업이므로, 신드롬 디코딩 문제는 코드 기반 암호가 양자내성을 가질 수 있게 해주는 기반이 된다. 코드 기반 암호는 행렬 연산을 사용하므로 암호화 및 복호화 연산 속도가 빠르다는 장점이 있지만 키 크기가 크다는 단점이 있다.

2.2.1 HQC

HQC (Hamming Quasi-Cyclic)는 hamming 코드와 랜덤한 Quasi-Cyclic (준순환) 코드를 사용하는 코드 기반 암호이다. Quasi-Cyclic은 일부 행렬에서 순환하는 관계가 성립하도록 하여 연산을 효율화시키는 것을 말하는데, 이를 활용하면 첫 번째 행만 저장하여도 전체 행렬을 알 수 있어 키 크기를 효율적으로 줄일 수 있다. <표 1>은 HQC의 공개키, 비밀키, 암호문의 크기 (bytes)를 나타낸다. 여기서 n은 $\mathbb{F}_2^n$을 의미하며, 2.2.2에서 언급한 HQC의 기약다항식에 의해 $n-1$이 각각의 hqc 인스턴스가 사용하는 바이너리 필드 크기가 된다. 4 라운드 후보인 코드 기반 암호에는 Classic McEliece, BIKE, HQC가 있는데, HQC는 BIKE보다 크고 McEliece보다 작은 파라미터 크기를 가진다. 많은 코드 기반 암호들이 Generator 행렬의 코드가 무작위성을 가질 수 있도록 변형하는 과정에서 취약점이 발생하여 해킹됐지만, HQC는 Generator 행렬 코드를 공개된 그대로 사용하기 때문에 코드 기반 구조의 문제가 없다고 볼 수 있으며 이는 HQC의 특징이며 큰 장점이다.

<표 1> 보안 강도별 HQC의 파라미터 크기 (bytes)

	pk	sk	ct	n
hqc-128	2,249	56	4,497	17,669
hqc-192	4,522	64	9,042	35,851
hqc-256	7,245	72	14,485	57,637

2.2.2 HQC의 PKE 구성

HQC의 PKE 버전은 크게 공개키 $pk = (h, s)$와 비밀키 $sk = (x, y)$를 생성하는 키 생성 단계와 메시지 m으로부터 암호문 $ct = (u, v)$을 생성하는 암호화 (인코딩) 단계, 비밀키를 통해 암호문으로부터 에러 e를 제거하여 메시지를 복구하는 복호화 (디코딩) 단계로 이루어진다. 바이너리 필드에서 사용되는 기약다항식은 $(X^n - 1)/(X - 1)$이며, 이때 n은 소수여야 한다. hqc-128의 경우 n은 17669가 되는데, 해당 식을 정리하면 $X^{n-1} + X^{n-2} + \cdots + X + 1$이 되므로 결론적으로 $\mathbb{F}_{2^{17668}}/(X^{17668} + X^{17667} + \cdots + X + 1)$이 사용된다. 또한 암호문 v를 구하는 인코딩 연산에서 메시지 m과 Generator 행렬 G를 곱할 때 shortened Reed-Solomon 코드가 사용되는데, 이때도 바이너리 필드 곱셈이 사용된다. 이때는 바이너리 필드 크기가 8이며, $\mathbb{F}_{2^8}/(X^8 + X^4 + X^3 + X^2 + 1)$의 기약 다항식이 사용된다. HQC는 인코딩할 때 더해지는 에러인 e가 매우 크기 때문에 디코딩 자체가 불가능하며, 비밀키를 가지고 있는 사용자만 e를 줄여서 디코딩을 쉽게 수행할 수 있게 된다. 디코딩은 확률에 따라 실패할 수도 있는데, HQC 논문에서 저자들은 상세하고 정확한 수학적 분석을 통해 실패 확률이 무시할 수 있는 수준으로 낮다는 것을 증명했다. 따라서 HQC는 높은 보안성을 제공한다고 볼 수 있다.

2.3 바이너리 필드 상에서의 산술 연산

바이너리 필드 상에서의 산술 연산은 일반적인 산술 연산과 다소 차이가 있다. 덧셈의 경우 캐리가 발생하지 않으므로 XOR을 사용하여 수행된다. 마찬가지로 곱셈도 캐리가 발생하지 않아 AND로 수행된다. 이때 곱셈 결과를 바이너리 필드 크기에 맞게 축소하는 모듈러 reduction 과정이 필요하며, 주어진 기약다항식을 통해 XOR를 사용하여 수행된다.

3. HQC PKE의 핵심연산에 대한 양자회로 최적 구현

본 장에서는 HQC PKE의 키 생성, 인코딩 단계에서 수행되는 핵심 연산인 바이너리 필드 산술과 shortened Reed-Solomon[3] 코드의 인코딩 연산에 대해 최적화된 양자회로 구현 기법을 설명한다.

3.1 키 생성 양자 회로 구현

키 생성은 동일한 바이너리 필드 상에 존재하는 x, y, h에 대하여 공개키 $s \leftarrow x + hy$를 구하는 부분이 핵심 연산으로, 바이너리 필드 덧셈 및 곱셈이 사용된다. hqc-128의 경우, $\mathbb{F}_{2^{17668}}/(X^{17668} + X^{17667} + \cdots + X + 1)$의 바이너리 필드에서 연산이 수행된다. 하지만 2^{17668}에서는 양자 시뮬레이션이 불가능하므로, 본 논문에서는 바이너리 필드를 12로 축소하여 양자회로를 구현하였으며, 해당 회로로 자원 추정을 수행한다. 기약다항식은 $\mathbb{F}$

<표 2> $\mathbb{F}_{2^8}/(X^8+X^4+X^3+X^2+1)$과 $\mathbb{F}_{2^{12}}/(X^{12}+X^{11}+\cdots+X+1)$ 산술 양자 회로 구현 결과

Field	Arithmetic	Qubits	CNOT gates	Toffoli gates	Toffoli depth	Full depth
$\mathbb{F}_{2^8}$	Multiplication	81	164	27	1	26
$\mathbb{F}_{2^{12}}$	Addition	24	12	-	-	1
	Multiplication	162	495	54	1	32

$_{2^{12}}/(X^{12}+X^{11}+\cdots+X+1)$을 사용한다. 덧셈 연산에 사용되는 XOR 연산은 양자회로에서 CNOT 게이트로 구현되므로, 필드 크기만큼의 CNOT 게이트를 통해 depth 1을 가지도록 구현할 수 있다. 하지만 AND 연산과 XOR 연산으로 수행되는 곱셈 연산은 바이너리 필드 산술 중 높은 계산 복잡도를 가지며, n의 크기를 가지는 바이너리 필드에서 Schoolbook 곱셈은 AND 연산이 n^2번 사용된다. 양자회로에서 AND 연산은 Toffoli 게이트로 구현되는데, Toffoli 게이트는 높은 구현 비용을 가진다. 따라서 바이너리 필드 상에서의 곱셈 연산을 최적화하기 위한 여러 연구가 수행되었으며[4-7], 본 논문에서는 가장 최근에 발표된, Toffoli-depth를 1로 최적화하는 양자 곱셈 기법[7]을 적용하여 바이너리 필드 곱셈을 구현하였다. 해당 곱셈 기법은 곱셈에서 다수의 덧셈을 추가로 수행하여 곱셈 복잡도를 줄인 카라추바 (Karatsuba) 알고리즘[8]을 재귀적으로 적용하며, 추가 큐비트 할당을 통해 곱셈 인자 간 종속성을 제거하여 필드 크기에 상관없이 Toffoli-depth를 1로 최적화하였다. 따라서 전체적인 depth를 매우 작게 형성하여 곱셈을 수행할 수 있다. 이전 연구[4-6]와 비교했을 때 큐비트 수가 가장 많이 필요하다는 점을 제외하면 Toffoli 게이트 수, Toffoli-depth, Full-depth 측면에서 가장 좋은 성능을 가진다. 큐비트 수와 depth는 trade-off 관계이지만, HQC의 경우 굉장히 큰 바이너리 필드에서 연산이 수행되기 때문에 필드 크기에 관계없이 항상 Toffoli-depth가 1인 [7]의 곱셈기는 본 구현에 가장 적합하다고 볼 수 있다.

3.2 인코딩 양자 회로 구현

인코딩은 동일한 바이너리 필드 상에 존재하는 r_1, r_2에 대하여 $u \leftarrow r_1 + hr_2$와 $v \leftarrow mG + sr_2 + e$를 구하는 연산으로 이루어진다. 이때 mG의 곱셈 연산에는 다른 연산들과 달리 shortened Reed-Solomon 코드가 사용되는데, 여기에서는 $\mathbb{F}_{2^8}/(X^8+X^4+X^3+X^2+1)$의 기약 다항식을 사용하여 필드 크기 8 상에서의 곱셈 연산이 수행된다. 이는 양자 시뮬레이션이 가능한 필드이므로 인코딩 양자 회로 구현에서 shortened Reed-Solomon 코드의 양자 회로 구현과 필드 크기를 축소한 나머지 바이너리 산술을 분리하여 구현 및 자원 측정을 수행하였다. mG 연산을 제외하고는 키 생성에서 구현한 바이너리 필드 상에서의 덧셈과 곱셈을 동일하게 사용한다. <표 2>는 각각 키생성과 인코딩에서 사용된 바이너리 필드 연산에 대해 양자 회로로 구현한 결과를 나타낸다.

[7]의 곱셈기를 사용함으로써 바이너리 필드 크기에 관계없이 Toffoli-depth가 1이 되는 것을 알 수 있다.

3.2.1 shortened Reed-Solomon 코드 양자 회로 구현

Algorithm 1 : shortened Reed-Solomon Encode

Input: 8-qubit array msg[K], RS_POLY[$G-1$], cdw[$N1$], gate_value[K], copy[$G-2$], ancilla qubits array ac[30]
Output: cdw

```
1: for i = 0 to G-1
2:   RS_POLY[i] ← CNOT8(RS_COEFS, RS_POLY[i])
3: for i = 0 to K
4:   gate_value[i] ← CNOT8(cdw[N1-K-1], gate_value[i])
5:   gate_value[i] ← CNOT8(msg[K-1-i], gate_value[i])
6:   for j = 0 to G-2
7:     copy[j] ← Copy_gate_value(gate_value[i], copy[j])
8:   tmp[0] ← Multiplication(gate_value[i], RS_POLY[0], ac[0])
9:   for j = 1 to G-1
10:    tmp[j] ← Multiplication(copy[j], RS_POLY[j], ac[j])
11:  for j = N1-K-1 to 0
12:    cdw[j] ← CNOT8(cdw[j-1], cdw[j])
13:    cdw[j] ← CNOT8(tmp[j], cdw[j])
14:  cdw[0] ← CNOT8(tmp[0], cdw[0])
15:  for j = 0 to G-2
16:    copy[j] ← Copy_gate_value(gate_value[i], copy[j])
17: for i = 0 to K
18:   cdw[i] ← CNOT8(msg[i], cdw[i+30])
19: return cdw
```

<알고리즘 1> shortened Reed-Solomon 인코딩 양자 구현

shortened Reed-Solomon 코드의 인코딩 양자회로는 <알고리즘 1>과 같다. 해당 알고리즘에서는 공개되어있는 RS-S1 다항식의 계수 행렬과 메시지 벡터를 곱하는 바이너리 필드 연산이 핵심이며, <알고리즘 1, line 8~10>이 곱셈 연산에 해당한다. 알고리즘에서 사용되는 상수 값은 각각 $K=16$, $G=31$, $N1=46$이다. <알고리즘 1, line 1~2>에서 다항식의 계수 행렬

값을 큐비트 배열로 나타내는데, 인코딩 연산에서는 31개의 계수 중 30개만 사용되므로 $G-1$까지만 수행하였다. <알고리즘 1, line 6~7>은 다음 단계에서 수행되는 30번의 바이너리 필드 곱셈을 병렬로 한꺼번에 처리하기 위해서 gate_value 값을 복사하는 부분이다. 이때 depth를 최소화하기 위해 29번의 값 복사 연산을 병렬로 나누어서 수행하도록 구현하였다. 이 작업을 통해 <알고리즘 1, line 8~10>에서 수행된 바이너리 필드 곱셈은 Toffoli-depth가 1로 최적화되며, 해당 연산이 총 16번 반복 수행되므로 Toffoli-depth는 16이 된다. 이때 tmp는 곱셈 연산이 in-place로 구현되었기 때문에 결과 큐비트를 담아두는 역할을 하는 변수이다. 또한 구현 값 복사 연산에 사용된 큐비트 배열은 <알고리즘 1, line 15~16>에서 reverse 연산으로 초기화하여 재활용하였다. <표 3>은 shortened Reed-Solomon 코드의 인코딩 연산을 양자 회로로 구현한 결과를 나타낸다. 앞서 설명했듯이 Toffoli-depth와 depth를 최적화하도록 구현하였기 때문에 사용된 게이트 수에 비해 낮은 depth를 가지는 것을 확인할 수 있다.

<표 3> shortened Reed-Solomon 코드 인코딩 양자 회로 구현 결과

shortened Reed-Solomon	Qubits	CNOT gates	Toffoli gates	Toffoli depth	Full depth
hqc-128	28,696	94,320	12,960	16	545

4. 결론

본 논문에서는 NIST 공모전의 4 라운드 후보 알고리즘인 HQC PKE 버전에 대한 키 생성 및 인코딩 연산 중 핵심 역할을 하는 바이너리 필드 산술과 shortened Reed-Solomon 코드의 인코딩 연산에 대한 최적화된 양자회로 구현을 제안하고 자원 추정을 수행하였다. 특히 양자 자원의 비용을 절감시키기 위해 바이너리 필드 상에서의 곱셈 연산을 최신 구현 기법을 적용하여 최적화하였다. 또한 shortened Reed-Solomon 코드의 경우 HQC에서 사용된 파라미터 그대로 양자회로로 구현하여 자원을 측정했다는 점에서 의의가 있다. 제시하는 양자회로를 통해 HQC의 보안강도 분석 연구에 기여할 수 있을 것으로 기대된다. 향후 Reed-Muller 코드 및 디코딩 단계의 핵심 연산까지 구현하여 HQC의 양자 회로 구현을 완성시키고자 한다. 또한 시뮬레이션이 가능한 범위를 조정하며 바이너리 필드를 최대한으로 확장할 계획이다.

5. Acknowledgment

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, 80%) and this work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, 20%).

참고문헌

[1] W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM review 41.2, pp. 303-332, 1999.

[2] M. Aguilar, A. Nicolas, B. Slim, B. Loïc, B. Olivier, B. Jurjen, D. Jean-Christophe, D. Arnaud, G. Philippe, L. Jérôme, P. Edoardo, R. Jean-Marc, V. Pascal, Z. Gilles, Hamming quasi-cyclic (HQC), NIST PQC Round 4, 2023.

[3] Aragon, N., Gaborit, P., Zémor, G.: Hqc-rmrs, an instantiation of the hqc encryption framework with a more efficient auxiliary error-correcting code (2020)

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

[5] S. Kepley and R. Steinwandt, Quantum circuits for F2^n-multiplication with subquadratic gate count, Quantum Information Processing, vol. 14, no. 7, pp. 2373 - 2386, 2015.

[6] I. Van Hoof, Space-efficient quantum multiplication of polynomials for binary fi-nite fields with subquadratic Toffoli gate count, arXiv preprint arXiv:1910.02849, 2019.

[7] K. Jang, W. Kim, S. Lim, Y. Kang, Y. Yan and H. Seo, Optimized Implementation of Quantum Binary Field Multiplication with Toffoli Depth One, International Conference on Information Security Applications, 2022.

[8] A. Karatsuba, Multiplication of multidigit numbers on automata, Soviet physics doklady, Vol. 7, pp. 595-596, 1963.