Classic McEliece 양자 회로 구현

한성대학교 오유진

NIST Post-Quantum Cryptography Standardization

- NIST 양자내성암호 표준화 공모전 진행 상황
  - 22년 7월 5일: 추가적으로 4 라운드 진행

NIST

Search CSRC

CSRC MENU

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

CSRC

UPDATES 2022

PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates

PQC Fourth Round Candidate Key-Establishment Mechanisms (KEMs)

The following candidate KEM algorithms will advance to the fourth round:

Public-Key Encryption/KEMs
BIKE
Classic McEliece
HQC
SIKE

코드기반암호

아이소지니기반암호

(Classic McEliece 표준화 예상)

Classic McEliece

Alice

Bob

Hey Bob! Establish a session key using Classic McEliece protocol

Generates Classic McEliece Key Pair

- Private key : $\Gamma = \{s, g(z), a_1, a_2, \ldots, a_n\}$
- Public key : T

Public key : T

Random Plaintext : e
Using $H = (I_{n-k} \mid T)$
Compute $C_0 = He^T$
Session Key : $K = H(1, e, C)$

Ciphertext: C

Decrypt C to obtain e
Session Key : $K = H(1, e, C)$

키 생성 양자 회로 구현

- 키 생성의 핵심 연산

  -> Binary Field 산술 : 덧셈, 제곱, 곱셈, 역치 연산

  mc Eliece348864 의 경우 $\mathrm{F}_{2^{12}}/(x^{12}+x^3+1)$

  그 외의 경우 $\mathrm{F}_{2^{13}}/(x^{13}+x^4+x^3+x+1)$ 상에서의 연산

- $\widetilde{H}=\{h_{i,j}\}$ over $\mathbb{F}_q$, $\boxed{h_{i,j}=a^{i-1}{}_j/g(a)}$ for $i=1,\dots,t$ and $j=1,\dots,n$

키 생성 양자 회로 구현

- 제곱 연산

LUP 분해를 기반으로 CNOT, Swap 게이트 사용하여 구현.

$$\boxed{0\ x_{11}\ 0\ x_{10}\ 0\ x_9\ 0\ x_8\ 0\ x_7\ 0\ x_6}\ 0\ x_5\ 0\ x_4\ 0\ x_3\ 0\ x_2\ 0\ x_1\ 0\ x_0$$

Modular reduction

- 모듈러 감소 후 행렬로 표현하여 LUP 분해(상삼각행렬, 하삼각행렬, Permutation 행렬).
- CNOT 게이트를 사용하여 연산하고 Swap 게이트로 permutation 진행
  → Swap 게이트는 큐비트의 인덱스를 변경하는 logical Swap이 가능함으로 비용 차지 X

키 생성 양자 회로 구현

- 곱셈연산

WISA'22 [2]의 toffoli depth가 1로 최적화되는 곱셈 기법 사용

Karatusba 알고리즘을 재귀적으로 사용한 Toffoli depth 가 1인 곱셈

[3] 은 기본적인 schoolbook 곱셈

Field	Arithmetic	Method	Qubits	Clifford gates	T gates	T-depth	Full depth
$F_{2^{12}}$	Multiplication	[3]	36	921	1,008	136	307
		[2]	162	761	378	4	37
$F_{2^{13}}$	Multiplication	[3]	42	1,110	1,183	148	333
		[2]	198	966	462	4	54

[3]이 큐비트를 많이 사용하지만 depth가 훨씬 감소

키 생성 양자 회로 구현

- 역치연산

Itoh-Tsujii 기반 역치 연산

$x^{-1} = x^{2^{12}-2}$ 곱셈과 제곱을 이용하여 연산 가능

앞서 구현한 곱셈과 제곱 활용

WISA'22 곱셈기를 사용한 역치연산

여러 곱셈 시에 ancilla 큐비트를 재사용할 수 있음

Field	Arithmetic	Qubits	Clifford gates	T gates	T-depth	Full depth
$F_{2^{12}}$	Inversion	402	4,758	1,890	20	194
$F_{2^{13}}$	Inversion	422	4,988	1,848	16	369

키 생성 양자 회로 구현

Field	Arithmetic	Method	Qubits	Clifford gates	T gates	T-depth	Full depth
$F_{2^{12}}$	Addition	-	24	12	-	-	1
	Squaring	-	12		-	-	2
	Multiplication	[3]	36	921	1,008	136	307
		[2]	162	761	378	4	37
	Inversion	[4]	402	4,758	1,890	20	194
$F_{2^{13}}$	Addition	-	26	13	-	-	1
	Squaring	-	13	7	-	-	2
	Multiplication	[3]	42	1,110	1,183	148	333
		[2]	198	966	462	4	54
	Inversion	[4]	422	4,988	1,848	16	369

인코딩 양자 회로 구현

- 공개키 H와 랜덤 벡터 e 의 행렬 벡터 곱셈 수행

$C_0 = He$ 공개키 H 는 사전에 정의되어 있는 값

가장 작은 파라미터의 공개키 (768 x 3844) 행렬에 대한 양자 시뮬레이션 불가능함

⟶ 축소된 행렬-벡터 곱셈 진행(8 x 16)

- Naïve (Out-of-place)
- PLU 분해(in-place)
- Quantum – Quantum 연산

Implementation	Method	Qubits	Clifford gates	T gates	T-depth	Full depth
Quantum-Quantum	Naive	152	784	896	92	147
Classical-Quantum	Naive	24	45	-	-	14
	LUP	16	37	-	-	13

인코딩 양자 회로 구현

- 양자 – 양자 구현

두 큐비트의 곱은 결국 둘 다 1일 때 저장할 큐비트에 1을 더해 주어야함
AND 연산 + XOR 연산

→ 즉, Toffoli 게이트를 사용.

$$\begin{bmatrix} 0&0&0&0&0&1&0&1 \\ 0&0&1&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0 \\ 1&0&1&1&1&0&0&0 \\ 0&0&0&1&1&1&0&1 \\ 0&0&1&0&1&1&0&1 \\ 1&1&1&1&1&1&0&1 \\ 1&1&0&0&1&1&1&0 \end{bmatrix} \text{x} \begin{bmatrix} 1\\1\\0\\0\\0\\0\\0\\0 \end{bmatrix} = \begin{bmatrix} 0\\0\\1\\1\\0\\0\\0\\0 \end{bmatrix}$$

```
def Encoding_1(eng, h, e, col, row):

    syndrome = eng.allocate_qureg(row)

    for i in range(row):
        # Quantum - Quantum
        h_e_mul(eng, h[(col*i):(col*i)+col], e, syndrome[row-1-i], col)

    return syndrome
```

인코딩 양자 회로 구현

- 클래식 – 양자. (Naïve)

H 값에 따라 CNOT 게이트(XOR) 를 결과 벡터에 수행. (out-of-place)

$$\begin{bmatrix} 0&0&0&0&0&1&0&1 \\ 0&0&1&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0 \\ 1&0&1&1&1&0&0&0 \\ 0&0&0&1&1&1&0&1 \\ 0&0&1&0&1&1&0&1 \\ 1&1&1&1&1&1&0&1 \\ 1&1&0&0&1&1&1&0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

H 값이 1일 때만 결과 큐비트와 CNOT 연산

```
def Encoding_2(eng, H, e, col, row):

    syndrome = eng.allocate_qureg(row)
    for i in range(row):
        for j in range(col):
            # Classic - Quantum
            if(H[col*i+j] == 1):
                CNOT | (e[col-1-j], syndrome[row-1-i])
```

인코딩 양자 회로 구현

- 클래식 – 양자 (LUP 분해)

행렬 H 를 LUP 분해 하여 벡터 e 에 결과 벡터가 연산되는 in-place 구현

LUP 분해

$$\begin{bmatrix} 0&0&0&0&0&1&0&1 \\ 0&0&1&0&1&0&1&0 \\ 1&0&0&0&0&1&1&0 \\ 1&0&1&1&1&0&0&0 \\ 0&0&0&1&1&1&0&1 \\ 0&0&1&0&1&1&0&1 \\ 1&1&1&1&1&1&0&1 \\ 1&1&0&0&1&1&1&0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
def Apply_U(eng, vector_e, row, col, U):
    for i in range(row):
        for j in range(col - 1 - i):
            if (U[(i * col) + 1 + i + j] == 1):
                CNOT | (vector_e[col - 2 - i - j], vector_e[col - 1 - i])
```

인코딩 양자 회로 구현

- 공개키 H와 랜덤 벡터 e 의 행렬 벡터 곱셈 수행

$C_0 = He$ 공개키 H 는 사전에 정의되어 있는 값

가장 작은 파라미터의 공개키 (768 x 3844) 행렬에 대한 양자 시뮬레이션 불가능함

⟶ 축소된 행렬-벡터 곱셈 진행(8 x 16)

- Naïve (Out-of-place)
- PLU 분해(in-place)
- Quantum – Quantum 연산

Implementation	Method	Qubits	Clifford gates	T gates	T-depth	Full depth
Quantum-Quantum	Naive	152	784	896	92	147
Classical-Quantum	Naive	24	45	-	-	14
	LUP	16	37	-	-	13

디코딩 양자 회로 구현

- 디코딩 알고리즘을 사용하여 C_0 으로부터 vector e 복구
  - **Berlekamp-Massey Decoding 알고리즘 사용**

- **Berlekamp-Massey Decoding 양자 회로 구현**
  - 핵심 연산은 $\mathbb{F}_{2^{12}}/(x^{12}+x^3+1)$, $\mathbb{F}_{2^{13}}/(x^{13}+x^4+x^3+\ x^1+1)$ 산술
  - 구현한 Multiplication, Inversion 활용

디코딩 양자 회로 구현

- **Berlekamp-Massey decoding 양자 회로**
  - mceliece344864
  - 곱셈 + 역치 연산 반복
  - WISA'22 곱셈 사용
  - Itoh-Tsujii 기반의 역치 사용
  - 큐비트 수를 줄이기 위해 곱셈과 역치 연산이 반복적으로 사용될 때 초기 할당한 대량의 보조 큐비트 (ac)를 연산 종료까지 재사용

Algorithm 3: The Berlekamp-Massey quantum circuit of Classic McEliece.

Input: 12-qubit b, 12-qubit array $T[t+1]$, $C[t+1], B[t+1], s[2t]$, ancilla qubits ac, $L = 0$ (classical)

Output: C

```
b ← X(b[0])
C[0] ← X(C[0][0])
B[1] ← X(B[1][0])
for N = 0 to 2t − 1 do
    d ← new 12-qubit allocation
    for i = 0 to min(N, t) do
        d ← MultiplicationXOR(C[i], s[N − i], ac)
    end for
    if (2L ≤ N) then
        for i = 0 to t do
            T[i] ← new 12-qubit allocation
            T[i] ← CNOT32(C[i], T[i])
        end for
    end if
    b^{-1} ← Inversion(b, ac)
    if (2L > N) then
        for i = 0 to t do
            C[i] ← MultiplicationXOR(f, B[i], ac)
        end for
    end if
    if (2L ≤ N) then
        for i = 0 to t do
            C[i] ← MultiplicationXOR(f, B[i], ac)
            L ← N + 1 − L (classical)
        end for
        for i = 0 to t do
            B[i] ← T[i]
        end for
        b = d (classical)
    end if
    for i = 0 to t − 1 do
        B[t − i] ← B[t − 1 − i]
    end for
    B[0] ← new 12-qubit allocation
end for
return C
```

Berlekamp-Massey decoding	Qubits	Clifford gates	T gates	T-depth	Full depth
mceliece344864	888,492	12,823,392	579,384	60,800	363,696

결론

- 본 논문에서는 NIST Round4 후보 알고리즘의 코드기반암호 중 하나인 Classic McEliece 핵심 연산들을 양자 회로로 구현.
- 특히 Berlekamp-Massey 디코딩 양자 회로는 최초로 제시하는 결과
- 향후 연구방향으로는 핵심 연산들을 기반으로 전체 양자 회로를 완성 시키는 것이 목표
- Classic McEliece 키 크기가 매우 큼에 따라 시뮬레이션이 가능한 범위를 조정해야함.

Q & A