Quantum Information Set Decoding (Error-Free)

장경배

https://youtu.be/_cXkz7ql9ds

한성대학교
HANSUNG UNIVERSITY

CryptoCraft LAB

QISD Efficiency

- Grover 알고리즘을 활용한 **코드기반암호** 분석의 경우 **대칭키 암호** 분석과 다름
- 대칭키 암호의 경우, ***n*-bit 키의 Search space**는 2^n, Grover 적용 시 $\sqrt{2^n}$
- 코드기반암호의 경우 **특정 무게의 n-bit 벡터를 찾는 경우**, 실제 Search space가 2^n이 아님

  → 이 경우 Grover가 비효율적으로 적용되어, **거의 동일한 성능**을 보여줌

McEliece parameters m, t	Workload Cryptanalysis (in binary operations) classic computer	quantum computer	Minimal number of Qubits	Quantum-computer bit security[7]
11, 32	2^{91}	2^{86}	25	80
11, 40	2^{98}	2^{94}	50	88
12, 22	2^{93}	2^{87}	29	80
12, 45	2^{140}	2^{133}	28	128

QISD Efficiency

- Grover가 **ISD(Prange)**에 완벽히 적용될 수 있음을 제시, **복잡도는 절반으로** 감소

Grover vs. McEliece

Daniel J. Bernstein *

Department of Computer Science (MC 152)
The University of Illinois at Chicago
Chicago, IL 60607–7053
djb@cr.yp.to

Abstract. This paper shows that quantum information-set-decoding attacks are asymptotically much faster than non-quantum information-set-decoding attacks.

Key words. code-based cryptography, post-quantum cryptography

Author(s)	Year	$\max_{0 \le R \le 1} \alpha(R, \omega_{\mathrm{GV}})$ to 4 dec. places
Prange [23]	1962	0.1207
Dumer [11]	1991	0.1164
MMT [18]	2011	0.1114
BJMM [4]	2012	0.1019
MO [19]	2015	0.0966

Author(s)	Year	Ingredients	$\max_{0 \le R \le 1} \alpha(R, \omega_{\mathrm{GV}})$ to 5 dec. places
Bernstein [5]	2010	Prange+Grover	0.06035
This paper	2017	Shamir-Schroeppel+Grover+QuantumWalk	0.05970
This paper	2017	MMT+"1+1=0"+Grover+QuantumWalk	0.05869

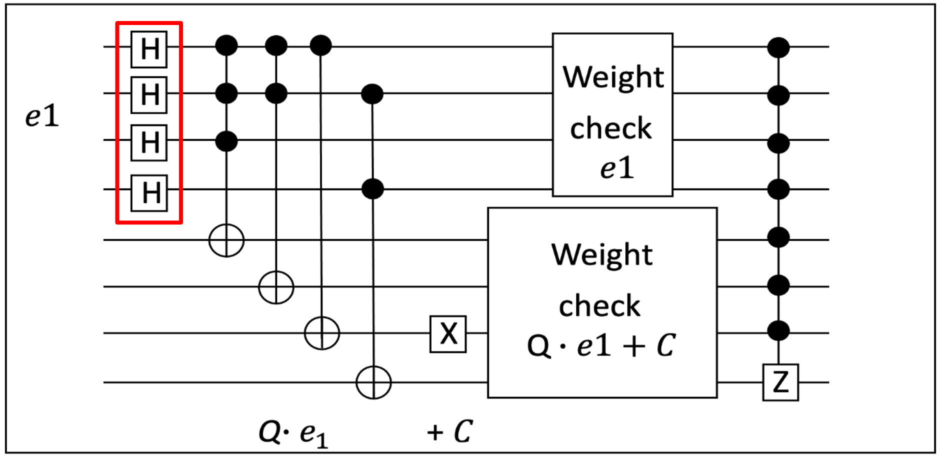
QISD Efficiency

- **Error-free information set** (Bernstein)

  In this paper I will be satisfied with a limited class of $\boxed{b\text{-bit-to-1-bit}}$ circuits,

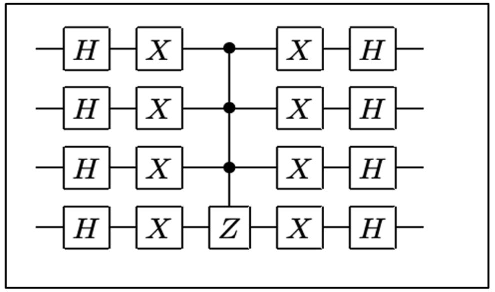
  Hamming weight

- 현재 회로에서는 고려 X



+

Reverse



Diffusion operator

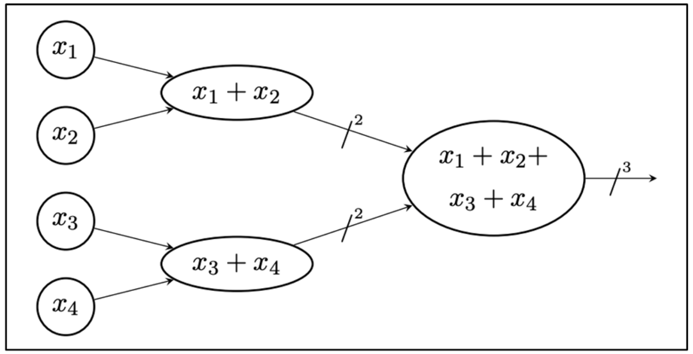
$\underbrace{\hspace{10em}}$

n 번 반복하여 관측 확률 증가

(Grover Search space → e1-qubit)

QISD Efficiency

- **Weight check** 에 대부분의 양자 자원이 사용됨



- **Weight check 한번**에 대한 비용이며, 추가로 **Grover iteration의 배수만큼** 더 소모됨

```
Gate counts:
    Allocate : 6978
    CCX : 12158
    CX : 24318
    Deallocate : 6978
    Measure : 2720

Depth : 28215.
```

QISD Efficiency

- 실제 Classic McEliece 파라미터에 대한 QISD

3.1 Parameter set kem/mceliece348864

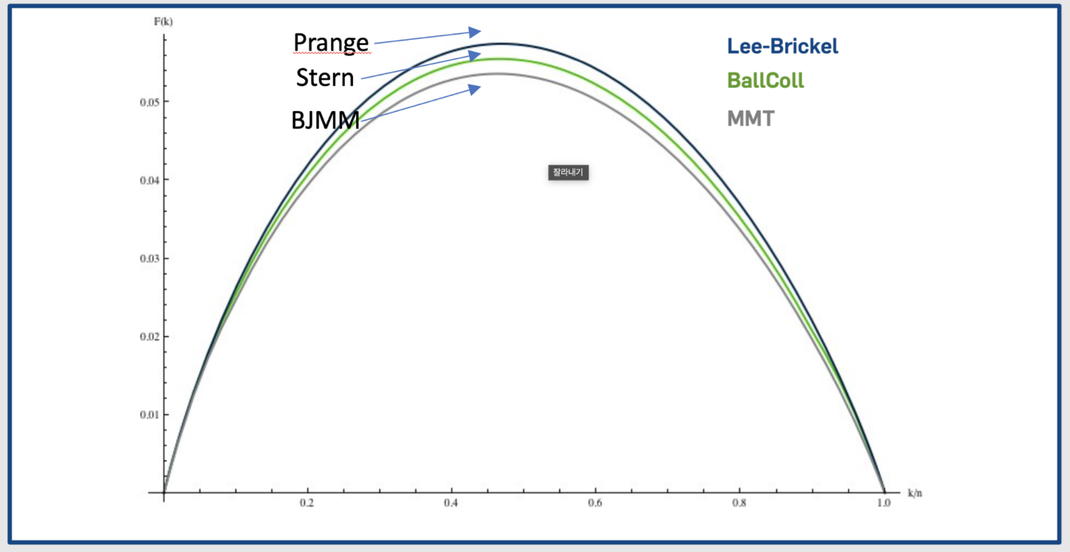
KEM with $m = 12$, $\boxed{n = 3488,}$ $t = 64$. Field polynomials $f(z) = z^{12} + z^3 + 1$ and $F(y) = y^{64} + y^3 + y + z$. This parameter set is **proposed and implemented** in this submission.

```
def QISD(eng):

    # Goppa code size : (768 x 3488)
    # Information set size : (768 x 2720)

    e = eng.allocate_qureg(2720)
    syndrome = eng.allocate_qureg(768)
    temp = eng.allocate_qureg(2040)
    temp2 = eng.allocate_qureg(576)
    carry = eng.allocate_qureg(682)
    carry2 = eng.allocate_qureg(191)

    c0 = eng.allocate_qubit()
```

Weight 계산에서 사용 되는 추가 큐빗

QISD Efficiency

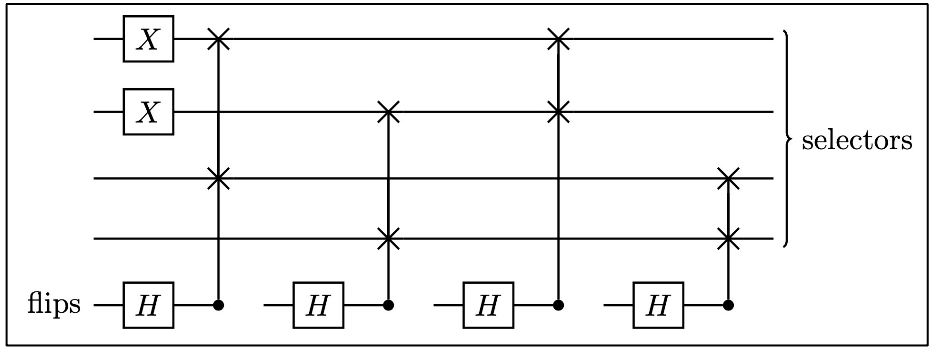
- 다양한 버전의 Infomration Set Decoding들이 있지만, 큰 **성능 차이 x**



- Quantum을 적용하였을 때 자원 측면에서의 효율성?
  - **Lee-Brickel**은 Weight check가 2번인 반면, **Prange**에서는 1번

Grover’s Algorithm + Guessing Phase

- 특정 Hamming weight를 고려하는 중첩 상태의 Input 설계



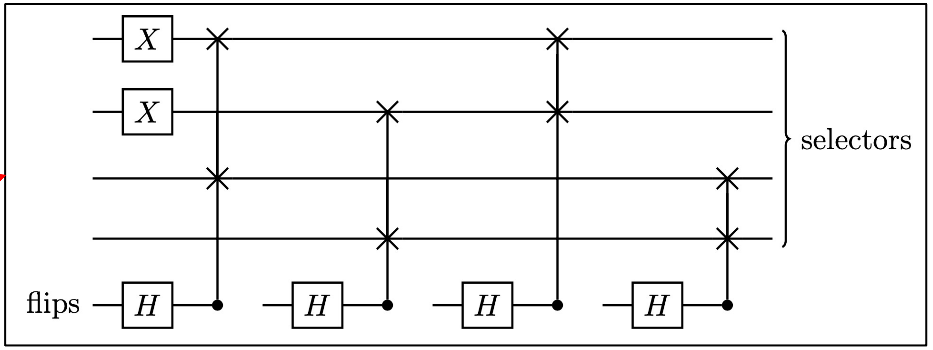
< 중첩 상태의 4-qubit vector, **Hamming weight = 2** >

- Search space 감소

$2^4 \rightarrow$ 1100
1010
1001
0110
0101
0011

Grover’s Algorithm + Guessing Phase

```
def ISD(eng):

    vector = eng.allocate_qureg(4)  # vector e
    flips = eng.allocate_qureg(4)  # vector e
    target = eng.allocate_qureg(4)  # syndrome s

    X | vector[0]
    X | vector[1] Hamming weight
    All(H) | flips

    for i in range(1): Iteration 횟수
        with Compute(eng):
            with Control(eng, flips[0]):
                Swap | (vector[0], vector[2])
            with Control(eng, flips[1]):
                Swap | (vector[1], vector[3])
            with Control(eng, flips[2]):
                Swap | (vector[0], vector[1])
            with Control(eng, flips[3]):
                Swap | (vector[2], vector[3])

            CNOT | (vector[0], target[2])
            CNOT | (vector[1], target[2])

            CNOT | (vector[2], target[1])
            CNOT | (vector[3], target[1])

            CNOT | (vector[2], target[2])

            X | target[0]
            X | target[1]
            X | target[3]       # answer list : 1000, 0100, 0011,

        with Control(eng, target[0:-1]):
            Z | target[-1]

        Uncompute(eng)
```



임의로 **0010**을 찾는 Oracle 설계, **Hamming weight 고려**

Index 순서 (0123)

Grover’s Algorithm + Guessing Phase

- 중첩 상태의 vector를 결정하는 건, **flips 큐비트**
  - flips 큐비트를 대상으로하여 Diffusion operator 수행
- 3개의 후보 중, Solution은 **0011**
- 4-qubit vector를 search, **반복 횟수는 1**

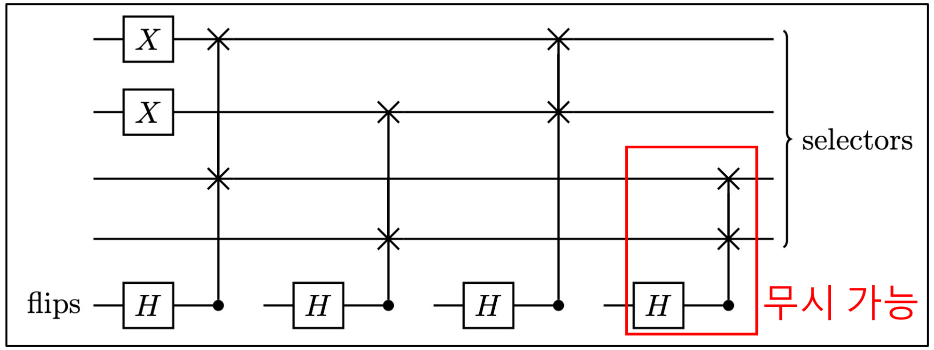
```python
def ISD(eng):

    vector = eng.allocate_qureg(4)  # vector e
    flips = eng.allocate_qureg(4)  # vector e
    target = eng.allocate_qureg(4)  # syndrome s

    X | vector[0]
    X | vector[1]
    All(H) | flips

    for i in range(1):
        with Compute(eng):
            with Control(eng, flips[0]):
                Swap | (vector[0], vector[2])
            with Control(eng, flips[1]):
                Swap | (vector[1], vector[3])
            with Control(eng, flips[2]):
                Swap | (vector[0], vector[1])
            with Control(eng, flips[3]):
                Swap | (vector[2], vector[3])

            CNOT | (vector[0], target[2])
            CNOT | (vector[1], target[2])

            CNOT | (vector[2], target[1])
            CNOT | (vector[3], target[1])

            CNOT | (vector[2], target[2])

            X | target[0]
            X | target[1]
            X | target[3]       # answer list : 1000, 0100, 0011,

        with Control(eng, target[0:-1]):
            Z | target[-1]

        Uncompute(eng)

        with Compute(eng):
            All(H) | flips
            All(X) | flips

        with Control(eng, flips[0:-1]):
            Z | flips[-1]

        Uncompute(eng)

        with Control(eng, flips[0]):
            Swap | (vector[0], vector[2])
        with Control(eng, flips[1]):
            Swap | (vector[1], vector[3])
        with Control(eng, flips[2]):
            Swap | (vector[0], vector[1])
        with Control(eng, flips[3]):
            Swap | (vector[2], vector[3])

    All(Measure) | vector
    All(Measure) | flips
```

Oracle

Diffusion operator

Grover’s Algorithm + Guessing Phase

- Hamming Weight(2)에 따른 Flips 최적화 가능,

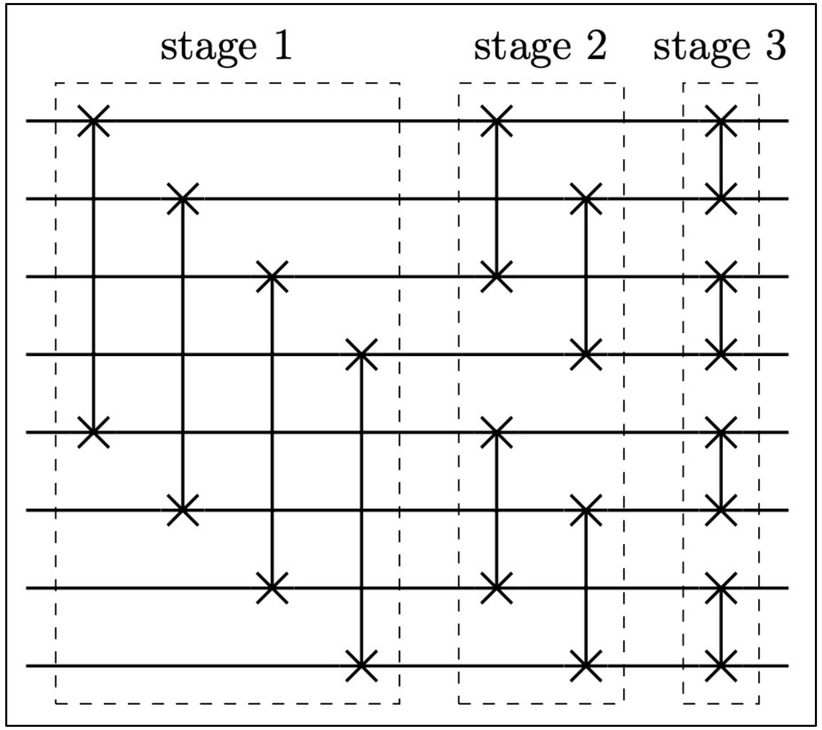


- 4-bit 벡터에서 Hamming weight가 2인 벡터들

1100
1010
1001
0110
0101
0011

Quantum ISD + Guessing Phase

- **6 x 8 의 Goppa code**에 대한 **QISD**
- **8-bit 벡터**를 대상으로 QISD**, Weight 는 2**



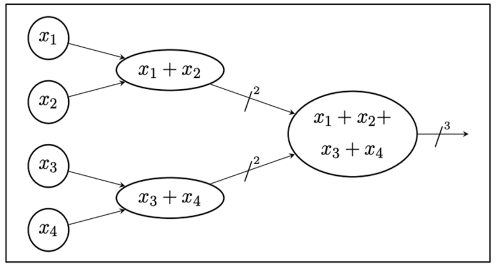

```python
def ISD(eng):

    mat = eng.allocate_qureg(8)  # vector e, weight=2
    target = eng.allocate_qureg(6)  # syndrome s
    flips = eng.allocate_qureg(10)  # weight result

    # n = 7  # optimal iteration number : pi/4 * arcsin(N^(1/2)) - (1/2)

    print('Answer e = 0 0 0 0 1 1 0 0')
    X | mat[0]
    X | mat[1]

    All(H) | flips
```

Conclusion

- **Guessing phase를 같이 고려**하면, ISD에 Grover search 알고리즘이 **완벽히 적용**될 수 있음

- Hamming Weight에 따른 **flips 부분 최적화 가능**

- **Weight check**에 필요한 자원들을 사용하지 않을 수 있음




```
def QISD(eng):

    # Goppa code size : (768 x 3488)
    # Information set size : (768 x 2720)

    e = eng.allocate_qureg(2720)
    syndrome = eng.allocate_qureg(768)
    temp = eng.allocate_qureg(2040)
    temp2 = eng.allocate_qureg(576)
    carry = eng.allocate_qureg(682)
    carry2 = eng.allocate_qureg(191)

    c0 = eng.allocate_qubit()
```

Weight 계산에서 사용 되는 추가 큐빗

- 더 알아봐야 함 → **완벽히 적용, 복잡도, flips 설계 및 최적화**

감사합니다