Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.3, Jun. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.3.323

해시함수 LSH 양자 회로 최적화를 통한 그루버 알고리즘 적용 자원 추정*

송 경 주,[1†] 장 경 배,[1] 서 화 정[2‡]
[1,2]한성대학교 (대학원생, 교수)

Resource Eestimation of Grover Algorithm through Hash Function LSH Quantum Circuit Optimization*

Gyeong-ju Song,[1†] Kyung-bae Jang,[1] Hwa-jeong Seo[2‡]
[1,2]Hansung University (Graduate student, Professor)

요 약

최근에는 양자 컴퓨터의 빠른 연산의 장점이 알려지면서 큐비트를 활용한 양자회로에 대한 관심이 높아지고 있다. 그루버 알고리즘은 n-bit의 보안 레벨의 대칭키 암호와 해시 함수를 $n/2$-bit 보안 레벨까지 낮출 수 있는 양자 알고리즘이다. 그루버 알고리즘은 양자 컴퓨터상에서 동작하기 때문에 적용 대상이 되는 대칭키 암호와 해시함수는 양자 회로로 구현되어야 한다. 이러한 연구 동기로, 최근 들어 대칭키 암호 또는 해시 함수를 양자 회로로 구현하는 연구들이 활발히 수행되고 있다. 하지만 현재는 큐비트의 수가 제한적인 상황으로 최소한의 큐비트 개수로 구현하는 것에 관심을 가지고 효율적인 구현을 목표로 하고 있다. 본 논문에서는 국산 해시함수 LSH 구현에 큐빗 재활용, 사전 연산을 통해 사용 큐빗 수를 줄였다. 또한, Mix, Final 함수와 같은 핵심 연산들을 IBM에서 제공하는 양자 프로그래밍 툴인 ProjectQ를 사용하여 양자회로로 효율적으로 구현하였고 이에 필요한 양자 자원들을 평가하였다.

ABSTRACT

Recently, the advantages of high-speed arithmetic in quantum computers have been known, and interest in quantum circuits utilizing qubits has increased. The Grover algorithm is a quantum algorithm that can reduce n-bit security level symmetric key cryptography and hash functions to $n/2$-bit security level. Since the Grover algorithm work on quantum computers, the symmetric cryptographic technique and hash function to be applied must be implemented in a quantum circuit. This is the motivation for these studies, and recently, research on implementing symmetric cryptographic technique and hash functions in quantum circuits has been actively conducted. However, at present, in a situation where the number of qubits is limited, we are interested in implementing with the minimum number of qubits and aim for efficient implementation. In this paper, the domestic hash function LSH is efficiently implemented using qubits recycling and pre-computation. Also, major operations such as Mix and Final were efficiently implemented as quantum circuits using ProjectQ, a quantum programming tool provided by IBM, and the quantum resources required for this were evaluated.

Keywords: Grover algorithm, quantum cryptography, LSH hash function, quantum, quantum circuit

Received(04. 12. 2021), Modified(05. 20. 2021), Accepted(05. 20. 2021)
* 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (〈Q|Crypton〉, No.2019-0-00033, 미래컴퓨팅 환경에 대비한 계산 복잡도 기반 암호 안전성 검증 기술개발).
* 본 논문은 2021년도 한국정보보호학회 영남지부 학술대회에 발표한 우수논문을 개선 및 확장한 것임
† 주저자, thdrudwn98@gmail.com
‡ 교신저자, hwajeong84@gmail.com(Corresponding author)

I. 서 론

최근 양자 컴퓨터의 빠른 연산의 장점이 알려지면서 큐비트를 통해 동시 연산할 수 있는 양자 회로에 대한 관심이 높아지고 있다.

양자 알고리즘에는 여러 가지가 있는데 그중 그루버 알고리즘은 n-bit 의 보안레벨을 가지는 대칭키 알고리즘과 해시 함수에 대하여 $n/2$-bit 보안레벨까지 낮출 수 있는 가장 효과적인 양자 공격 방법이다〔1〕. 양자 알고리즘인 그루버 알고리즘을 활용하기 위해서는 적용 대상 암호나 해시 함수를 양자 회로로 구현해야만 한다. 현재 양자 컴퓨터는 아직 초기 개발 단계이기 때문에 사용 가능한 큐빗이 매우 적고, 복잡한 양자 연산에서 생기는 오류를 제어하기 매우 까다롭다. 때문에 양자 회로를 최적화 구현하여 효율적으로 자원을 사용하는 것이 중요하다. 이러한 연구 동기에 따라 최근에는 대칭키 암호와 해시 함수를 양자 회로로 최적화 구현하는 연구들이 많이 진행되고 있다. 〔2〕는 블록암호 AES를 양자 회로로 구현하여 그루버 알고리즘을 적용하는데 필요한 양자 자원을 추정하였고, 〔3,4〕는 앞선 연구보다 최적화 된 AES 양자 회로를 제안하였다. 〔5〕는 미국 NSA(National Security Agency)에서 개발한 경량 블록암호 SIMON을 양자 회로로 구현하였으며 〔6〕에서는 SPECK을 양자 회로로 구현하였다. 마지막으로 〔7〕은 국산 경량 블록 암호 HIGHT, LEA, CHAM을 양자 회로로 구현 하였다.

양자 회로 구현 시에는 사용되는 큐빗의 수를 줄이는 것이 매우 중요하다. 대규모 큐빗의 양자 컴퓨터가 아직 개발되지 않았기 때문에 구현에 사용되는 큐빗의 수가 실제 동작이 가능해지는 시기와 연관이 있다. 본 논문에서는 해시함수 LSH를 양자회로로 구현함에 있어 최소한의 큐빗을 사용하는 것을 목표로 하였으며 이와 같은 조건하에서 양자 게이트 수를 가능한 줄이는 방식으로 최적화하여 구현하였다.

해시함수는 임의의 길이의 비트 열을 입력 받아 고정 길이의 해시 값을 출력하는 함수이다. 해시함수 LSH는 2014년 국가보안기술연구소에서 개발한 해시함수로, 높은 안전성과 우수한 효율성을 제공한다. 안전성으로는 충돌 쌍 공격, 역상 공격 등 해시 함수와 관련한 공격에 대하여 안전하게 설계되었으며, 국외 전문 연구기관인 벨기에 COSIC(Computer Security and Industrial Cryptography) 연구소로부터 안전성에 대한 객관적 검증을 받았다. 또한, 국제 표준해시함수인 SHA-2와 SHA-3와 비교하여 효율성 3배 이상의 우수한 성능을 갖는다〔8〕.

LSH 는 메시지를 입력으로 하여 고정길이 비트의 해시값을 출력하는 해시함수이다〔10〕. 비트 워드 단위로 동작하며 n 비트 출력 값을 가지는 해시함수 LSH-8w-n (family : LSH-256-224, LSH-256-256, LSH-512-224, LSH-512-256, LSH-512-384, LSH-512-512) 으로 구성된 해시함수 군(Hash Function Family)이다. 이때, w는 32 또는 64 이며 n은 1과 8w 사이의 정수이다.

본 논문에서는 LSH-256-n (n = 224, 256), LSH-512-m (m=224, 256, 384, 512) 총 6개의 LSH을 모두 양자 회로로 구현하여 이에 그루버 알고리즘을 적용하기 위한 양자 자원들을 추정하였다.

LSH는 초기화, 압축, 완료 단계로 진행되며 초기화 단계에서는 입력 메시지의 패딩을 통해 메시지를 확장시킨다. 압축 단계에서는 초기화 단계에서 확장시킨 메시지와 연결변수 CV를 압축 함수 CF의 입력 값으로 하여 연결변수 값을 업데이트 하며 진행한다. 압축 단계에서의 함수는 메시지 확장(MsgExp), 메시지 덧셈(MsgAdd), 섞음(Mix), 워드 단위 순환 함수(WordPerm) 이렇게 네 가지가 있다. 마지막으로 완료 함수에서 최종 해시값을 출력한다.

제안하는 양자회로는 연결변수 값을 업데이트하기 위해 쓰이는 큐빗을 줄이기 위해 사전에 연산해 놓은 단계 상수 값에 따라 진행하도록 하였으며 메시지 확장 함수를 통해 (N_s+1)개의 16워드 배열 $M_j^{(i)}\,(0 \le j \le N_s)$을 위한 큐빗을 생성하지 않고 이전의 메시지 큐빗을 재활용하며 라운드를 진행하도록 하였다.

II. 관련연구

2.1 quantum ripple-carry addition〔9〕

본 논문에서 제안하는 LSH 해시 함수의 덧셈연산은 최소한의 큐빗을 사용하기 위해 해당 논문의 향상된 ripple-carry addition 양자 회로를 사용하였다. 해당 논문에서 제안하는 새로운 ripple-carry addtion 양자 회로는 덧셈의 대상이 되는 2개의 입력 값 중 한 개의 입력 값에 덧셈 결과를 저장하기

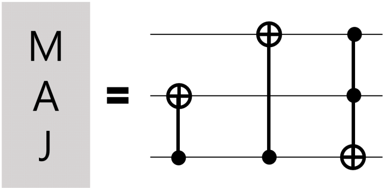

Fig. 1. MAJ quantum circuit

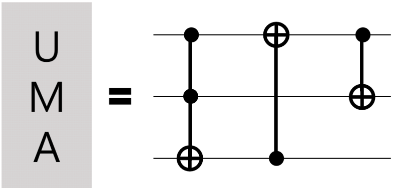

Fig. 2. UMA quantum circuit

때문에 덧셈 결과를 저장하기 위한 추가 큐빗이 필요하지 않으며 carry 값을 저장하기 위한 1큐빗만을 추가로 사용하므로 큐빗을 최소화 할 수 있다. ripple-carry addition 양자 회로에는 Fig 1, 2와 같은 MAJ, UMA 양자 게이트가 사용된다. MAJ는 두 개의 CNOT 게이트와 한 개의 Toffoli 게이트가 사용된다. UMA는 2개의 CNOT 게이트와 한 개의 Toffoli 게이트가 사용된다.

Fig. 3. 은 6-qubit ripple-carry addition 양자 회로이다. MAJ 및 UMA 의 조합으로 구성되어 있다. 이때, s_i는 덧셈 대상 a_i와 b_i의 덧셈 합 $s_i = a_i + b_i$ 을 나타낸다.

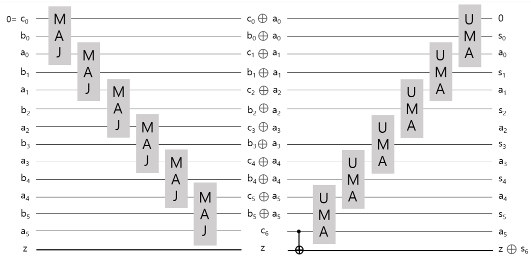

Fig. 3. Ripple-carry adder (6-qubit)

2.2 해시함수 LSH[10]

LSH 는 메시지를 입력으로 하여 특정 길이의 해시값을 출력하는 해시함수이다. Fig. 4.와 같이 전체 구조를 가지며 초기화, 압축, 완료 이렇게 총 세

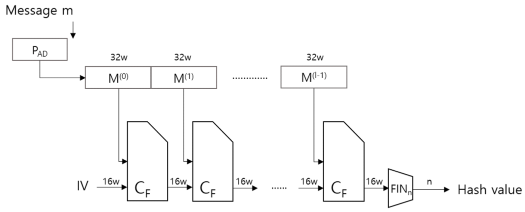

Fig. 4. LSH hash function

단계를 거쳐 해시 값을 출력한다.

2.2.1 완료 함수 FIN_n

완료 함수 $FIN_n : W_{16} \rightarrow \{0,1\}n$ 은 압축 과정 결과를 통해 업데이트 된 최종 연결 변수 $CV^{(t)} = (CV^{(t)}[0], \cdots, CV^{(t)}[15])$에 적용하여 LSH-256-n (n = 224, 256), LSH-512-m (m = 224, 256, 384, 512) 에서 n, m 비트 길이의 해시값 h를 생성하는 함수이다. $H = (H[0], \cdots, H[7])$ 를 8 워드 배열, $h_b = (h_b[0], \cdots, h_b[w-1])$를 w바이트 배열이라고 하면, 완료함수 FIN_n 는 수식 (1) 을 수행하여 최종 해시값을 출력한다.

$$
\begin{aligned}
& H[i] = CV^{(t)}[i] \oplus CV^{(t)}[i+8] \quad (0 \le i \le 7), \\
& h_b[s] = H[(8s/w)]^{\gg (8s \bmod w)}[7:0] \\
& (0 \le s \le (w-1)), \\
& h = (h_b[0] \mid\mid \cdots \mid\mid h_b[w-1])_{[0:n-1]}
\end{aligned}
\tag{1}
$$

2.2.2 압축 함수

압축 함수의 입력값 중 메시지 블록 M(i)는 메시지 확장 함수 MsgExp를 통해 (N_s+1) 개의 16워드 배열인 $M_j^{(i)} (0 \le j \le N_s)$로 확장되어 단계를 진행한다. Fig. 5.는 압축 함수의 진행 과정이다. 확장된 메시지 블록 M(i) 과 16 워드 배열 크기의 임시 변수 $T = (T[0], \cdots, T[15])$에 초기값을 $CV^{(i)}$로 할당한 값을 단계 함수 의 입력함수로 하여 $M_j^{(i)}$를 처리하면서 T 를 업데이트한다. 단계 함수를 거치며 T에 업데이트된 값은 MsgAdd 함수를 통해 처리된 후 $(i+1)$번째 연결 변수 $CV^{(i+1)}$에 입력된다.

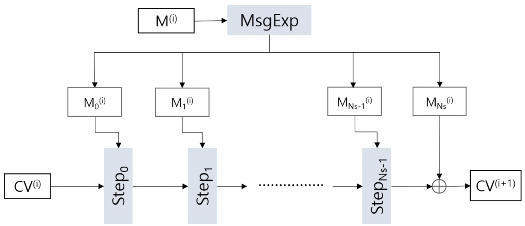

Fig. 5. Compression function

2.3 그루버 알고리즘

그루버 알고리즘은 정렬되지 않은 N개의 데이터 셋에서 특정 데이터를 $O(\sqrt{N})$번 만에 찾아내는 양자 알고리즘이다. 그루버 알고리즘은 대칭키 암호에 대한 키 서치 또는 해시 함수에 pre-image 공격에 사용될 수 있으며 크게 Oracle과 Diffusion operator로 구성된다. Oracle 내부에는 구현한 해시 함수 또는 대칭키 암호화 양자 회로가 자리한다. Oracle에 입력되는 input 큐빗들은 Hadamard 게이트를 통과하여 중첩 상태가 된다. 따라서 모든 input 큐빗에 대한 모든 경우의 값들을 표현하게 되고 해시함수의 경우, 양자 회로를 거쳐 모든 output 해시 값들이 확률로써 존재하게 된다. 이 중 찾으려 하는 해시 값과 일치하는 해시 값을 가진 큐빗의 상태를 찾고 부호를 반전 시키는 Z 게이트를 수행한다.

Diffusion operator는 부호가 반전된 큐빗의 amplitude를 증폭시킴으로써 관측 확률을 증가시킨다. 최종적으로 그루버 알고리즘은 Oracle과 Diffusion operator를 반복 수행하여 높은 확률로 솔루션을 관측할 수 있다.

그루버 알고리즘에 사용되는 양자 자원들은 대부분 Oracle에 영향을 받으며, Oracle의 주요 모듈은 해시 함수 양자회로 또는 대칭키 암호화 회로가 될 수 있다. 따라서 그루버 알고리즘을 활용한 키 서치 또는 해시 함수 pre-image 공격에는 대상 암호 알고리즘을 양자 회로로 효율적으로 구현하는 것이 중요하다.

III. 제안기법

본 논문에서는 LSH-256-n (n = 224, 256), LSH-512-m (m=224, 256, 384, 512) 총 6개의 모든 LSH 해시함수를 대상으로 양자 회로를 구현하였다.

LSH-256-n 에서는 32bit를 1word 단위로 진행하며 입력 메시지를 패딩 한 1024비트 평문 M을 위한 1024큐빗, 라운드 함수의 입력 값이 되는 연결변수 CV를 위한 512 큐빗, ripple-carry 덧셈의 캐리 값을 저장하기 위한 1큐빗이 사용되었다.

LSH-512-m 에서는 64bit를 1word 단위로 진행하며 입력 메시지를 패딩 한 2048비트 평문 M을 위한 2048큐빗, 라운드 함수의 입력 값이 되는 연결변수 CV를 위한 1024큐빗, ripple-carry 덧셈의 캐리 값을 저장하기 위한 1큐빗이 사용되었다.

LSH-256-n, LSH-512-m 의 전체적인 동작 과정은 같으며 각 동작 비트 단위 수와 단계상수, 최종 해시값 출력 방식이 다르다는 점을 주의하여 작성하였다.

전체 메시지 $M_j(0 \le j \le N_s)$를 확장 한 후 단계함수를 진행하기 위해서는 확장한 메시지 전체를 저장하기 위한 추가 큐빗이 필요하다. 이러한 추가 큐빗 할당을 줄이기 위해서 16워드 배열의 메시지 블록을 하나씩 확장하여 단계함수를 진행하였다. 이러한 방식으로 이전 단계함수를 마친 후 사용 완료한 메시지 큐빗을 이후 메시지 확장에서 재사용하도록 하였다. 예를 들어 메시지를 확장하는 식 $M_j^{(i)}[l] \leftarrow M_{j-1}^{(i)}[l] \boxplus M_{j-2}^{(i)}[\tau(l)]$ 을 통해 세 번째 메시지 블록 M_2 를 확장할 때 M_1과 M_0의 덧셈연산을 통해 생성된다. 이때, $\tau(l)$은 치환표를 통해 치환된 값이다[10]. 하나의 메시지 블록 단위로 메시지를 확장 및 단계함수를 진행하면 메시지 블록 M_2를 사용할 때는 M_0과 M_1이 이미 사용되었으므로 M_2를 확장할 때 M_1과 M_0의 덧셈연산 값을 M_0에 저장하여 진행 할 수 있다. 이미 사용된 큐빗 M_0을 재사용하였기 때문에 M_2를 저장하기 위한 추가 큐빗을 할당할 필요 없이 메시지를 확장하고 압축함수를 진행할 수 있다. 또한, 라운드를 진행할 때 Mix 함수에서 사용하는 단계상수 SC의 값을 업데이트하기 위한 큐빗을 따로 할당하지 않고 연산식 $SC_j[l] \leftarrow SC_{j-1}[l] \boxplus SC_{j-1}[l]^{\lll 8} (0 \le l \le 7, 1 \le j \le (N_s - 1))$을 통해 classical computer로 사전에 계산 값을 계산하고 상수로서 사용하여 XOR연산을 진행하도록 구현하였다. 제안하는 기법을 통해 앞서 언급한 큐빗 외의 추가 큐빗을 할당하지 않고 최소한의 큐빗을 사용하는 조건에서 양자게

이트를 가능한 줄이는 방식으로 양자회로를 설계하였다.

3.1 LSH 양자회로

LSH-256-n 에서는 1024비트의 메시지를 1 word (word = 32 bit) 씩 16개의 $M[t]\,(0 \le t \le 15)$ 로 나눠 Step(단계함수)을 진행하며 LSH-512-m 에서는 2048비트의 메시지를 1word (word = 64 bit) 씩 16개의 $M[t]\,(0 \le t \le 15)$ 로 나눠 단계함수를 진행한다.

M은 업데이트 되어 매 라운드에 사용되는데 제안하는 기법에서는 M의 업데이트 값을 위한 큐빗을 추가로 할당하지 않고 이전 라운드에서 사용된 M에 새로운 값을 생성함으로써 큐빗을 절약하였다.

연결변수 $T[i], T[i+8]\,(0 \le i \le 7)$ 는 MsgExp, Mix, WordPerm 함수를 진행하며 T를 갱신하고 갱신한 값을 최종적으로 Final함수를 통해 해시 값을 얻는다. 양자회로를 이용한 Mix 함수 구현은 Fig. 6.과 같다.

Fig. 6.은 Mix함수의 양자회로 구현이다. 두 개의 워드 쌍 $T[i], T[i+8]\,(0 \le i \le 7)$ 으로 한번의 Mix 함수를 동작하며 한 라운드 당 총 8번의 Mix 함수가 사용된다. 각 2, 5, 7 번째 라인 a, b, c rotation 함수에서의 순환 값은 각 word 의 단위 수 (32, 64 bit) 및 단계함수 $Step_j$의 j 값에 따라 순환 상수값이 다르다. 단계함수 $Step_j$의 j 값이 짝수/홀수 에 따라 a, b 로테이션의 비트 순환량 α_j, β_j 가 다르며 c 로테이션의 비트 순환량 γ은 단계 변수 $T[k]\,(k = 8, 9, 10, 11, 12)$의 k값에 따라 바뀌기 때문에 각 경우를 주의하여 순환량 α_j, β_j, γ 을 미리 설정해 두고 Swap 연산을 통해 비트를 로테이션 시켰다.

Fig 6 : LSH Quantum circuit implementation of Mix

```
Input: T[i], T[i+8], SC[i] (0≤i≤7)
Output: T = {T[0] … T[15]}
1: ripple_carry_add (T[i], T[i+8])
2: a_rotation(T[i])
3: Applying X gate to T[i] according to SC[i]
4: ripple_carry_add(T[i], T[i+8])
5: b_rotation(T[i+8])
6: ripple_carry_add(T[i+8], T[i])
7: c_rotation(T[i+8])
8: return T = {T[0] … T[15]}
```

Fig. 6. LSH Quantum circuit implementation of Mix

Fig. 6.의 3 라인은 CNOT-gate를 통해 미리 연산해둔 단계 상수 SC와 T를 XOR 연산하였다. Mix 함수를 통해 얻은 $T = \{T[0], \cdots, T[15]\}$ 은 WordPerm 함수에서 Table. 1.에 맞춰 Swap을 통해 치환된다. 비트 로테이션 및 WordPerm 함수에서 사용된 Swap 연산들은 서로 비트 위치만을 주는 연산이므로 별도의 게이트 비용이 들지 않는다.

LSH 해시 함수에서는 사용되는 각 라운드에 대한 SC[i] 값들은 초기 단계상수의 연산을 통해 업데이트 된다. 따라서 모든 라운드에서 사용할 각 단계상수 SC[i]를 $SC_j[l] \leftarrow SC_{j-1}[l] \boxplus SC_{j-1}[l]^{\lll 8}$ $(0 \le l \le 7, 1 \le j \le (N_s - 1))$ 식을 통해 사전 연산하여 제안하는 기법에서는 해당 라운드의 SC[i]의 비트 값이 1인 위치에 따라 T[i] 위치의 큐비트에 X 게이트를 수행한다. 때문에 SC[i]값을 할당하기 위한 큐빗들이 전혀 사용되지 않았으며 기존의 SC[i] 값을 업데이트하기 위한 양자 게이트 또한 사용되지 않았다.

LSH-256-n 에서는 $N_s\,(N_s = 26)$번의 단계함수 $Step_{N_s}$을 진행하며 LSH-512-m 에서는 $N_s\,(N_s = 28)$번의 $Step_{N_s}$을 진행한다. Step을 진행하며 업데이트 된 CV를 이용하여 Final 함수에서 최종 해시 값을 얻을 수 있다. 양자 회로를 이용한 LSH 의 Final 함수에서는 각 LSH 별로 수식 (1) 에 맞춰 결과를 얻는다.

Fig. 7. 는 수식 (1) 에 맞춰 양자 회로로 작성한 LSH-256-256 의 Final 함수이다.

Fig. 7. 의 2번째 라인은 $CV_{k+8}, CV_k\,(0 \le k \le 7)$ 쌍으로 짝지어 총 7번의 CNOT-gate를 수행하여 XOR 연산한 값 $CV_0, \cdots, CV_7$ 을 얻는다. Fig. 7.의 4번째 라인은

Table 1. WordPerm Substitution table$I$

	0	1	2	3	4	5	6	7
$\sigma(I)$	6	4	5	7	12	15	14	13
I	8	9	10	11	12	13	14	15
$\sigma(I)$	2	0	1	3	8	11	10	9

Fig 7 : LSH-256-256 Quantum circuit implementation of Final

```
Input: CV[i], CV[i+8] (0≤i≤7)
Output: h = { h[0] , … , h[256] }
for k = 0 to 7
    for i = 0 to 31
        CNOT (CV_{K+8}[i], CV_K[i])
for k = 0 to 7
    for i = 7 to 0
        print (CV_K[i])
    for i = 15 to 8
        print (CV_K[i])
    for i = 23 to 16
        print (CV_K[i])
    for i = 31 to 24
        print (CV_K[i])
```

Fig. 7. LSH-256-256 Quantum circuit implementation of Final

$CV_k(0 \le k \le 7)$를 차례로 오른쪽으로 8bit 씩 로테이션 하면서 $CV_k[i](0 \le k \le 7, 7 \le i \le 0)$의 8비트 출력을 하는 연산 대신 출력 비트 자리를 옮겨 8비트씩 출력하도록 하여 로테이션 연산에 필요한 Swap을 사용하지 않고 값을 출력하였다. Final 함수를 통해 출력한 최종 비트 h는 곧 최종 해시 값이 된다.

IV. 구현 결과 및 분석

제안하는 LSH 양자 회로의 테스트 벡터 검증을 위해 IBM ProjectQ의 ClassicalSimultor를 활용하였다. ProjectQ 컴파일러 중 하나인 ClassicalSimultor는 X, CNOT, Toffoli 게이트와 같은 단순 양자 게이트들만을 허용하며, 이를 통해 대규모 큐빗의 양자 회로의 구현 적합성을 검증할 수 있다. 사용 된 양자 자원은 또 다른 컴파일러인 ResourceCounter를 활용하여 분석하였다. 제안하는 LSH 그리고 기존 연구 결과인 SHA2, SHA3 양자 회로 구현에 필요한 자원들은 Table. 2.에 나타나있다. 회로 뎁스는 연산의 시작부터 종료까지를 측정한 수치이며 뎁스가 짧을수록 양자 회로의 수행 속도는 빨라진다.

LSH는 메시지를 입력 받으면 블록 메시지 길이의 배수만큼 패딩하여 각 단계를 진행한다. 본 논문에서는 1024, 2048 블록 메시지 길이의 1배수인 각

Table 2. Quantum resources for LSH and SHA

	Qubits	Toffoli gates	CNOT gates	X gates	Depth
LSH-256-224	1,537	63,488	145,152	1,536	210,051
LSH-512-224	3,073	139,104	312,832	7,663	421,851
LSH-512-256	3,073	139,104	312,832	7,696	421,851
LSH-512-384	3,073	139,104	312,832	7,668	421,850
LSH-512-512	3,073	139,104	312,832	7,680	421,852
LSH-256-256	1,537	63,488	145,152	3,492	210,049
SHA2-256[14]	2,402	57,184	534,272	-	528,768
SHA3-256[14]	3,200	84,480	33,269,760	85	10,128

1024, 2048 비트 메시지 블록을 해시할 때 사용되는 양자 자원을 평가하였다. 이때, 양자회로는 큐빗의 수를 최소로 하고 이와 같은 조건에서 게이트 수를 가능한 줄이는 것에 중점을 두고 설계하였다. SHA2는 512비트의 블록을 256비트 해시 값으로 출력하는 것을 기준으로 양자자원을 평가한 것이며 SHA3는 스펀지 구조를 가지므로 입력 길이에 상관없이 256비트 해시 값을 출력하는 것을 기준으로 양자자원을 측정한 것이다.

LSH-256-256 이 SHA2-256 보다 두 배 길이의 블록메시지를 해시 시킨 것을 감안하고 LSH-256-256 과 SHA2-256, SHA3-256의 양자회로 자원과 비교해 볼 때 LSH가 SHA보다 적은 양의 게이트와 큐빗을 사용하였다. 최소한의 큐빗을 사용하기 위해 해시함수 LSH 양자회로 에서는 단계를 거치며 M 값을 업데이트 하는데 기존 M에 할당된 큐빗을 재활용하여 총 사용 큐빗을 줄였으며 압축단계에서 XOR 연산하기 위한 각 라운드마다의 SC 값을 게이트를 통해 업데이트 하는 것이 아니라 모두 사전에 연산하고 이에 따라 X 게이트를 T에 수행하였다. 또한, Mix, Final 등의 연산들을 효율적인 방법으로 구현함으로써 LSH의 양자회로에 사용되는 큐빗 수와 양자 게이트 수를 최소화 하였다.

V. 결 론

본 논문에서는 기존의 국산 해시 함수

LSH-256-n (n = 224, 256)과 LSH-512-m (m = 224, 256, 384, 512)을 양자 회로를 이용하여 최적화 구현한 후, 최종적으로 그루버 알고리즘의 Oracle에서 해시함수의 output인 해시 값을 통해 input 값을 찾기 위한 양자 자원을 추정해 보았으며 이는 양자 컴퓨터의 공격에 대한 안전성의 지표로 활용할 수 있다. 제시하는 해시함수 LSH 양자 회로 구현은 잠재적으로 그루버 알고리즘에 효과적으로 적용될 수 있을 것이라 기대된다.

References

〔1〕 Grover, L.K, “A fast quantum mechanical algorithm for database search.” Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 212 219, July. 1996.

〔2〕 Grassl. M, Langenberg. B, Roetteler. M, and Steinwandt. R, “Applying Grover’s algorithm to AES: quantum resource estimates. Post-Quantum Cryptography.” Springer, pp. 29 - 43, Feb. 2016.

〔3〕 Langenberg, Brandon, Hai Pham, and Rainer Steinwandt. ″Reducing the cost of implementing the advanced encryption standard as a quantum circuit.″ IEEE Transactions on Quantum Engineering 1: 1-12, Feb. 2020

〔4〕 Jaques. S, Naehrig. M, Roetteler. M, and Virdia. F, “Implementing Grover oracles for quantum key search on AES and LowMC.” Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, pp. 280 - 310, 2020.

〔5〕 Anand, Ravi, Arpita Maitra, and Sourav Mukhopadhyay. ″Grover on $$\, SIMON\, $$ SIMON.″ Quantum Information Processing 19.9: 1-17, Seb. 2020.

〔6〕 K.B. Jang, S.J. Choi, H.D. Kwon, and H.J. Seo, “Grover on SPECK : Quantum Resource Estimates.” ePrint Archive, Report 640(2020), 2020.

〔7〕 K.B. Jang, S.J. Choi, H.D. Kwon, H.J. Seo, H.J. Kim, J.H. Park, and H.J. Seo, “Grover on Korean Block Ciphers”, Appl. Sci. 10.18: 6407. 2020.

〔8〕 Korea Cryptography Forum, ″High Speed Hash Function LSH″, 2015 LSH Implementation Contest, Korea Cryptography Forum, 2015

〔9〕 Cuccaro, Steven A., et al. ″A new quantum ripple-carry addition circuit.″ arXiv preprint quant-ph/0410184 (2004).

〔10〕 Kim DC., Hong D., Lee JK., Kim WH., and Kwon D. (2015) LSH: A New Fast Secure Hash Function Family. In: Lee J., Kim J. (eds) Information Security and Cryptology – ICISC 2014. ICISC 2014.

〔11〕 Chen, Zhao-Yun, et al. ″64-qubit quantum circuit simulation.″ Science Bulletin 63.15 (2018): 964-971.

〔12〕 Diao, Zijian, Zubairy, M. Suhail and Chen, Goong. ″A Quantum Circuit Design for Grover's Algorithm″ Zeitschrift für Naturforschung A, vol. 57, no. 8, 2002, pp. 701-708.

〔13〕 Maslov, Dmitri, et al. ″Quantum circuit simplification and level compaction.″ IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27.3 (2008): 436-444

〔14〕 Amy, Matthew, et al. ″Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3.″ International Conference on Selected Areas in Cryptography. Springer, Cham, Aug. 2016.

〈저자소개〉

송 경 주 (Gyeong-ju Song) 학생회원
2021년 3월: 한성대학교 IT융합공학부 졸업
2021년 3월~현재: 한성대학교 IT융합공학부 석사과정
〈관심분야〉 정보보호, 암호, 양자컴퓨터

장 경 배 (Kyung-bae Jang) 학생회원
2019년 3월: 한성대학교 IT응용시스템공학부 졸업
2021년 3월: 한성대학교 IT융합공학부 석사
2021년 3월~현재: 한성대학교 정보컴퓨터공학과 박사과정
〈관심분야〉 정보보호, 암호, 양자컴퓨터

서 화 정 (Hwa-jeong Seo) 종신회원
2010년 3월: 부산대학교 컴퓨터공학과 졸업
2012년 3월: 부산대학교 컴퓨터공학과 석사
2016년 3월: 부산대학교 컴퓨터공학과 박사
2016년~2017년: 싱가포르 과학기술청 연구원
2019년~현재: 한성대학교 IT융합공학부 조교수
〈관심분야〉 정보보호, 암호화 구현, IoT