해시함수 LSH 양자 회로 최적화를 통한 그루버 알고리즘 적용 자원 추정

송경주* 장경배* 서화정**†

*한성대학교 (대학원생)

**한성대학교 (교수)

Resource Estimation Applying Groover Algorithm through Hash Function LSH Quantum Circuit Implementation

Gyeong-Ju Song*, Kyung-Bae Jang*, Hwa-Jeong Seo**†

*Hansung University(Graduate student)

**Hansung University(Professor)

요 약

그루버 알고리즘은 n-bit의 보안 레벨의 대칭키 암호와 해시 함수를 $2/n$-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. 본 논문에서는 국산 해시함수 LSH를 양자 회로로 최적화하여 구현하였다. 큐빗 재활용, 사전 연산 그리고 Mix, Final 함수와 같은 핵심 연산들을 효율적으로 구현하여 국산 해시함수 LSH를 양자 회로로 설계하는데 필요한 양자 자원을 평가하였다.

I. 서론

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 효율적인 자원을 사용하는 것이 매우 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 최적화된 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.

본 논문에서는 해시함수 LSH를 양자회로로 최적화하여 구현하였다. 해시함수는 임의의 길이의 비트 열을 입력 받아 고정 길이의 해시 값을 출력하는 함수이다. LSH는 2014년 국가보안기술연구소에서 개발한 해시함수로, 높은 안전성과 우수한 효율성을 제공한다. 안정성으로는 충돌쌍 공격, 역상 공격 등 해시 함수와 관련한 공격에 대하여 안전하게 설계되었으며, 국외 전문 연구기관인 벨기에 COSIC(Computer Security and Industrial Cryptography) 연구소로부터 안정성에 대한 객관적 검증을 받았다. 효율성은

국제 표준해시함수인 SHA-2와 SHA-3와 비교하여 3배 이상의 우수한 성능을 갖는다.[8]

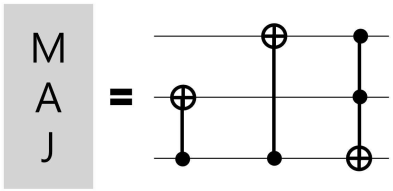
w 비트 워드 단위로 동작하며 n 비트 출력 값을 가지는 해시함수 LSH-8w-n으로 구성된 해시함수 군(Hash Function Family)이다. LSH는 w는 32 또는 64 이며 n은 1과 $8w$ 사이의 정수이다. 본 논문에서는 LSH-256/256을 양자 회로로 최적화 구현하여 이에 그루버 알고리즘을 적용하기 위한 양자 자원들을 추정하였다.

II. 관련연구

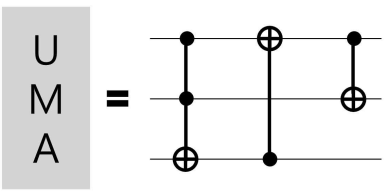
2.1 quantum ripple-carry addition[9]

본 논문에서는 ripple-carry addition 양자 회로를 사용하였다.. rippl-carry addtion 양자 회로는 덧셈의 대상이 되는 2개의 입력 값 중 한 개의 입력 값에 덧셈 결과를 저장하기 때문에 큐빗을 최소화 할 수 있다.

그림 1은 MAJ 양자 회로이다. 두 개의 CNOT 게이트와 한 개의 Toffoli 게이트가 사용된다.

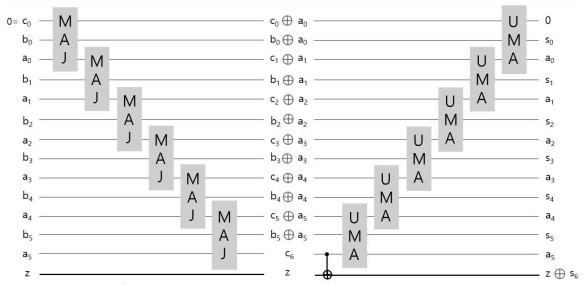


(그림 1) MAJ quantum circuit



(그림 2) UMA quantum circuit

그림 2 는 UMA 양자 회로이다. 각 큐비트에서 동일한 계산하는 함수를 두가지 제공한다. 2개의 CNOT 게이트와 한 개의 Toffoli 게이트가 사용된다.

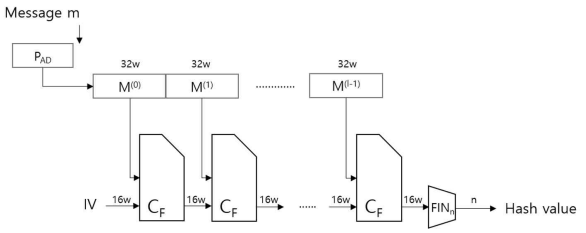


(그림 3) Ripple-carry adder (6-qubit)

그림 3은 6-qubit ripple-carry addtion 양자 회로이다. MAJ 및 UMA 의 조합으로 구성되어 있다.

2.2 해시함수 LSH[10]

LSH를 구성하는 각 해시함수는 그림 4와 같은 전체 구조를 가지며, 입력 메시지에 대해 초기화, 압축, 완료 세 가지 단계를 거쳐 해시값을 출력한다.



(그림 4) LSH hash function

2.2.1 완료 함수 FIN_n

완료 함수 FIN_n : W16 → {0,1}n은 압축 과정의 결과로 생성된 마지막 연결 변수값 CV(t) = (CV(t)[0], … , CV(t)[15])에 적용되어 n 비트 길이의 해시값 h를 생성한다. H = (H[0], … , H[7])를 8 워드 배열, hb = (hb[0], … , hb[w-1])는 w 바이트 배열이라고 하면, 함수 FIN_n은 수식 1을 수행한다.

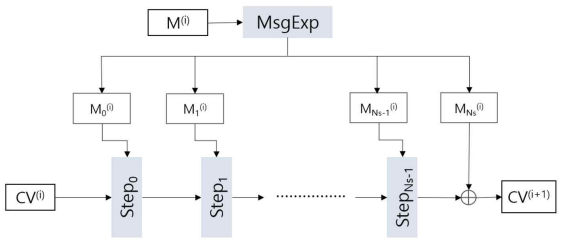
$$H[i] = CV^{(t)}[i] \oplus CV^{(t)}[i+8] \quad (0 \le i \le 7),$$
$$h_b[s] = H[\,(8s/w)\,]^{\gg (8s \bmod w)}[7:0] \quad (0 \le s \le (w-1)),$$
$$h = (h_b[0] \mid\mid \cdots \mid\mid h_b[w-1])_{[0:n-1]}$$

(수식 1) FIN_n function

2.2.2 압축 함수

압축 함수의 입력값 중 메시지 블록 M(i)는 메시지 확장 함수 MsgExp를 거쳐 (Ns + 1) 개의 16 워드 배열 M_j(i) (0 ≤ j ≤ Ns)로 확장된다. 이어서 16 워드 배열 크기의 임시 변수 T = (T[0], … , T[15])에 초기값을 CV(i)로 할당한 후, 순차적으로 단계 함수 Stepj를 통해

$M_j(i)$를 처리하면서 T를 갱신한다. 단계 함수를 거쳐 T에 저장된 값은 MsgAdd 함수를 통해 처리된 후 (i + 1) 번째 단계 변수 CV(i+1)에 입력된다.



(그림 5) Compression function

III. 제안기법

LSH-256/256를 대상으로 양자 회로를 구현하였으며, 입력 메시지를 패딩 한 1024비트 평문 M을 위한 1024큐빗, 라운드 함수의 입력 값이 되는 CV를 위한 1024 큐빗, ripple-carry 덧셈의 캐리 값을 저장하기 위한 1큐빗, 마지막으로 최종 해시 값을 저장하는 256큐빗이 사용되었다.

초기 할당된 메시지 M을 재활용 하였으며 라운드 함수에서 사용되는 SC값을 위한 큐빗을 할당하지 않고 사전 연산 값에 따라 라운드 함수를 구현하였다. 이를 통해 SC를 위한 큐빗들을 할당하지 않았으며 SC를 업데이트 하는 양자 게이트 또한 사용되지 않았다. 제안하는 기법을 통해 앞서 언급한 큐빗 외 추가 큐빗을 할당하지 않았으며, 양자 게이트의 사용 또한 최적화 하였다.

LSH-256/256에서는 1024비트의 메시지를 1 word (word=32bit) 씩 16개의 M[t] ($0 \le t \le 15$)로 나눠 Step(단계함수)을 진행한다. M은 업데이트 되어 매 라운드에 사용되는데 제안하는 기법에서는 M의 업데이트 값을 위한 큐빗을 할당하지 않고 기존 사용된 M에 새로운 값을 생성함으로써 큐빗을 절약하였다. 연결 변수 T[i], T[i+8] ($0 \le i \le 7$)는 MsgExp, Mix, WordPerm 함수를 진행하며 T를 갱신하고 최종적으로 Final함수를 통해 해시 값을 얻는다. 양자회로를 이용한 Mix 함수 구현은 알고리즘 1과 같다.

Algorithm 1 : Quantum circuit implementation of Mix

Input: T[i], T[i+8], SC[i] ($0 \le i \le 7$)
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]}

(알고리즘 1) Quantum circuit implementation of Mix

두 개의 워드 쌍 T[i], T[i+8] ($0 \le i \le 7$) 으로 한번의 Mix를 동작하며 총 8번의 Mix 함수가 사용된다. 알고리즘 1의 2, 5, 7 번째 라인 a, b, c 로테이션 함수에서의 순환 값은 각 Stepj 의 j값이 짝수/홀수 에 따라 a, b 로테이션의 비트 순환량 αj, βj 가 다르며 c 로테이션의 비트 순환량은 단계 변수의 값 T[k] (k = 8, 9, 10, 11, 12) 에 따라 바뀌기 때문에 각 경우를 주의하여 순환량을 설정해 두고 Swap 연산을 통해 비트를 로테이션 시켰다.

알고리즘 1의 3 라인은 CNOT-gate를 통해 SC(단계 상수)와 T를 XOR 연산하였다. Mix 함수를 통해 얻은 T = {T[0] ⋯ T[15]} 은 WordPerm 함수에서 Swap을 통해 치환된다. 비트 로테이션 및 WordPerm에서 사용된 Swap 연산은 서로 비트 위치만을 주는 연산이므로 별도의 게이트 비용이 들지 않는다. 제안하는 기법에서는 모든 라운드에 대한 SC[i]를 사전 연산하여 이에 따라 라운드 마다 T[i]에 X 게이트를 수행한다. 때문에 큐빗이 사용되지 않았으며 기존의 SC[i] 값을 업데이트하기 위한 양자 게이트를 전혀 사용하지 않았다.

Ns (Ns = 26)번의 Step을 마치고 얻은 CV를 이용하여 Final 함수에서 최종 해시 값을 얻을 수 있다. 양자 회로를 이용한 Final 함수 구현은 알고리즘 2와 같다.

Algorithm 2 : Quantum circuit implementation of Final

Input: CV[i], CV[i+8] ($0 \le i \le 7$)
Output: h = { h[0] , ⋯ , h[256] }
1: **for** k = 0 to 7
2: **for** i = 0 to 31
3: **CNOT** (CV_{K+8}[i], CV_K[i])
4: **for** k = 0 to 7

```
5:    for i = 0 to 7
6:        CNOT (CV_K[7-i], h_K[i])
7:    for i = 0 to 0
8:        for j = 0 to 31
9:            Swap (CV_K[i], CV_K[i+1])
10:   for i = 0 to 7
11:       CNOT (CV_K[7-i], h[8*(k*4)+i])
12:   for i = 0 to 0
13:       for j = 0 to 31
14:           Swap (CV_K[i], CV_K[i+1])
15:   for i = 0 to 7
16:       CNOT (CV_K[7-i], h[16*(k*4)+i])
17:   for i = 0 to 0
18:       for j = 0 to 31
19:           Swap (CV_K[i], CV_K[i+1])
20:   for i = 0 to 7
21:       CNOT (CV_K[7-i], h[24*(k*4)+i])
```

(알고리즘 2) Quantum circuit implementation of Final

알고리즘 2의 2번째 줄은 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. $CV_K(0 \le k \le 7)$를 1비트씩 로테이션하며 $CV_K[7],\cdots,CV_K[0]$ 순서로 8비트씩을 0으로 세팅된 256비트 큐빗에 순서대로 저장한다. 이때 CNOT-gate를 이용한 XOR 연산을 통해 대입 연산을 대신한다. Final 함수를 통한 256개의 비트 h는 곧 해시 값이 된다.

IV. 구현 결과

LSH-256/256을 양자 회로로 구현하는데 필요한 자원은 표 1과 같다.

	Qubits	Toffoli gates	CNOT gates	X gates
LSH-256/256	1,918	63,488	145,408	3,495

(표 1) Quantum resources for LSH-256/256

LSH는 메시지를 입력 받으면 블록 메시지 길이의 배수만큼 패딩한다. 본 논문에서는 블록 메시지 길이의 1배수인 1024 비트 블록을 해시할 때 사용되는 양자 자원을 평가하였다.

M 값을 업데이트 하는데 있어 기존 M에 할당 된 큐빗을 재활용하였으며 SC 값을 XOR 연산하기 위해 라운드마다의 SC 값을 모두 사전에 연산하고 이에 따라 X 게이트를 T에 수행하였다. 또한 Mix, Final 등의 연산들을 최적화함으로써 LSH의 양자회로에 사용되는 큐빗 수와 양자 게이트 수를 최소화 하였다.

V. 결론

본 논문에서는 국산 해시 함수 LSH-256/256을 양자 회로로 최적화 구현하여, 최종적으로 필요한 자원을 추정하였다. 해시 함수를 양자 회로로 구현하여 그루버 알고리즘 오라클에 적용하기 위한 자원을 확인함으로써 양자 컴퓨터의 공격에 대한 안전성의 지표로 활용할 수 있다. 제시한 LSH 해시 함수의 양자 회로 구현을 잠재적으로 그루버 알고리즘에 활용할 수 있을 것이라 기대된다.

VI. Acknowledgment

이 논문은 부분적으로 2019년도 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) 그리고 이 성과는 부분적으로 2020년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2020R1F1A1048478).

[참고문헌]

[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, - 1996.

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

[3] Langenberg. B, Pham. H, Steinwandt. R, “Reducing the cost of implementing AES as a quantum circuit.” Technical report, Cryptology ePrint Archive, Report 2019/854, 2019.

[4] Jaques. S, Naehrig. M, Roetteler. M, 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, pp. 280 – 310, 2020.

[5] Anand, R.; Maitra, A.; Mukhopadhyay, S. “Grover on SIMON.” 2020.

[6] K.B. Jang, S.J. Choi, H.D. Kwon, H.J. Seo, “Grover on SPECK : Quantum Resource Estimates.” ePrint Archive, Report 2020/640, 2020.

[7] K.B. Jang, S.J. Choi, H.D. Kwon, H.J. Seo, H.J. Kim, J.H. Park, H.J. Seo, “Grover on Korean Block Ciphers”, Appl. Sci. 10, 6407. 2020.

[8] Korea Cryptography Forum, "High Speed Hash Function LSH", 2015 LSH Implementation Contest, Korea Cryptography Forum, 2015

[9] Cuccaro, Steven & Draper, Thomas & Kutin, Samuel & Moulton, David. (2004). A new quantum ripple-carry addition circuit.

[10] Kim DC., Hong D., Lee JK., Kim WH., 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.