정 보 보 호 학 회 지
제34권 제5호, 2024. 10

패스워드 크래킹 암호들에 대한 양자 컴퓨팅 기반 해킹 동향

송 경 주*, 서 화 정**

요 약

정보화 시대에는 개인정보들이 디지털 정보로 저장되며 이를 보호하기 위한 보안 기술들이 연구되고 있다. 패스워드와 같은 기밀 정보들의 보안을 유지하기 위해 개발된 암호 시스템들은 양자 컴퓨터의 등장으로 인해 기존 보안성을 보장하기 어렵다. 패스워드 알고리즘에 대한 해킹을 위해서는 타겟 암호가 양자컴퓨터 상에서 구현되어야 하며 이후에 양자 알고리즘인 그루버 알고리즘을 사용하여 암호 해킹을 수행하게 된다. 양자컴퓨팅 상에서 보다 패스워드 크래킹을 효율적으로 수행하기 위해 양자회로 최적화 구현 연구들이 진행되고 있다. 본 논문에서는 그 중 패스워드 크래킹 암호인 Argon2과 Scrypt에 대한 양자회로 구현동향을 살펴보고 각 양자회로의 최적화 구현 기법 및 양자자원 추정결과를 확인한다.

Ⅰ. 서 론

현대 사회에서는 많은 중요 정보가 디지털 정보로 저장되며 이러한 개인 정보 및 프라이버시를 지키는 것이 중요한 문제로 대두되고 있다. 인터넷 상에서는 사용자의 인증을 위해 패스워드를 사용하며 이를 보호하기 위한 연구들이 많이 진행되고 있다 [1][2]. 패스워드 보안을 유지하기 위해 개발된 다양한 암호화 기법들은 공격자로부터 사용자의 패스워드를 보호하는데 필수적인 역할을 한다. 특히, 패스워드를 단방향 함수로 설계된 해시함수를 통해 해시형태로 저장하는 방식은 보안성을 높일 수 있는 주요 방식 중 하나이다. 하지만 최근 양자컴퓨터의 발전은 인해 이러한 암호들의 보안성을 위협할 수 있는 잠재력을 가지고 있다. 양자 컴퓨터는 고전적인 컴퓨터와 달리 양자역학 특성을 통해 특정 문제에 대해 빠르게 해결할 수 있으며 이러한 특징으로 설계된 양자 알고리즘인 그루버 알고리즘은 대칭키 암호 및 해시함수에 대해 brute-force 공격 및 pre-image 공격을 가속화 한다.

본 논문에서는 패스워드 크래킹 암호들 중 Argon2[3] 및 Scrypt[4] 대한 양자 컴퓨팅을 통한 해킹 동향을 살펴본다. 양자 알고리즘을 이용해 패스워드 크래킹 암호를 공격하기 위해서는 타겟 암호에 대해 양자컴퓨터 상에서 동작하는 양자회로로 구현해야 한다. 암호의 양자 보안 강도 및 시기를 정확히 확인하기 위해서는 양자컴퓨터 상에서 효율적으로 구현하는 것이 필요하며 이와 같은 연구동기로 블록암호 및 해시함수를 양자회로로 효율적으로 구현하기 위한 연구들이 많이 진행되었다[5-8]. 구현된 타겟 암호의 양자회로는 그루버 알고리즘의 오라클에서 사용되며 공격에 필요한 자원을 추정하여 암호의 양자 보안 강도를 예측할 수 있다.

Ⅱ. 관련 연구

본 세션에서는 대칭키 암호에 대한 brute-force 공격에 사용되는 양자알고리즘인 그루버 알고리즘에 대해 설명한다.

2.1. 그루버 알고리즘

그루버 알고리즘은 1996년 Lov Grover가 제안한 양자 알고리즘으로 대칭키 암호에 대해 키를 다항 시간에 찾을 수 있도록 하여 위협이 된다[9]. 그루버 알고리즘은 크게 3단계로 진행된다: Initialization, Oracle, Difusion operation. Initialization 에서는 key

* 한성대학교 정보컴퓨터공학과 (대학원생, thdrudwn98@gmail.com)
** 한성대학교 융합보안학과 (부교수, hwajeong84@gmail.com)

의 모든 가능한 상태를 균등한 중첩(superposition) 상태로 초기화하며 Oracle은 특정 조건에 부합하는 결과를 구분하는 역할을 한다. 공격 대상이 되는 암호의 양자회로는 Oracle에 존재하며 반복을 통해 타겟 암호에 맞는 key를 찾을 수 있다고 알려졌다. Difusion operation에서는 확률적으로 찾은 정답 key의 확률 진폭을 증폭시킨다.

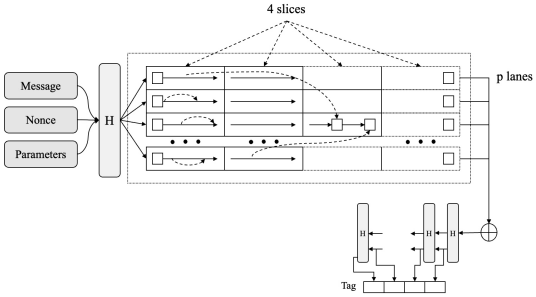
Ⅲ. 양자컴퓨팅을 통한 암호 해킹 동향

본 세션에서는 패스워드 크래킹 암호인 Argon2, Scrypt에 대한 양자회로 구현 동향을 살펴본다.

3.1. Argon2

[10]에서는 자격 증명 저장, 키 파생 또는 기타 애플리케이션을 해싱하는 데 사용할 수 있는 키 파생 알고리즘인 Argon2에 대한 양자회로를 제시했다. Argon2는 세가지 변형인 Argon2, Argon2d, Argon2i, Argon2id을 제공하며 [그림 1]과 같이 동작한다.

Argon2는 기본 입력(Primary inputs)과 보조 입력 또는 매개변수(Secondary inputs)라는 두 가지 유형의 입력을 가진다. 기본 입력으로는 메시지 P와 nonce S가 있으며 매개 변수로는 병렬 처리 정도를 나타내는 정수 p $(1 \le p \le 2^{24}-1)$, 바이트 값의 태그 길이 τ $(4 \le \tau \le 2^{32}-1)$, 메모리 크기 m $(8p \le \tau \le 2^{32}-1)$, 반복 횟수인 정수 t $(1 \le t \le 2^{32}-1)$, 버전 번호 v (1바이트, 0x13), 바이트 단위의 비밀값 K$(0 \le K \le 2^{32}-1)$, 바이트 단위의 관련 데이터 X $(0 \le X \le 2^{32}-1)$, Argon2의 유형 y (Argon2d:0, Argon2i:1, Argon2id:2)가 있다.. 각 입력은 p개의 lanes와 4개의 slices로 나뉘어져 동작한다.



〔그림 1〕 Argon2 프로세스 동작

3.1.1. Argon2 양자회로

해당 논문에서는 두 가지 최적화 관점: 큐비트 감소(큐비트 최적화), 깊이 감소(깊이 최적화)의 Argon2 양자회로를 제시한다. 큐비트 수 및 깊이는 현재 양자 컴퓨터 개발 단계에서 중요한 두 가지 요소이며 trade-off 관계를 가진다. 따라서 두 최적화 요소에 대한 양자회로를 나눠서 구현하였다.

큐비트 최적화된 양자회로에서는 회로 중간에 역연산을 통해 큐비트를 특정 연산 이전으로 되돌려 연산에서 사용한 후에도 다시 재사용할 수 있는 전략을 채택하였다. 논문에서는 이러한 작업을 위해 큐비트를 최소한의 계산으로 재사용할 수 있는 적절한 역연산 지점을 설정한다. 반면, 깊이 최적화된 양자회로에서는 역 연산을 사용하지 않고 ancilla 큐비트를 사용하여 큐비트 수를 늘리는 대신 계산 깊이을 최적화 하기 위한 전략을 채택하였다. 더 나아가 깊이 최적화 양자회로의 일부 구조에서 추가적인 ancilla 큐비트를 사용하여 병렬성을 늘려 깊이를 줄인다.

양자회로 내에서는 양자 덧셈이 수행되는데, ripple이라고 언급하는 (6n-2) 깊이의 덧셈기와 simple이라고 언급되는 (2n+3)깊이의 덧셈기를 큐비트 및 깊이 최적화 양자회로에 각각 적용한다. 따라서 사용된 덧셈기에 따라 각 최적화 회로 별 두가지 결과를 나타낸다. 두 최적화 양자회로에 공통적으로 채택된 기술들은 다음과 같다. Classic to Quantum 이라는 공통된 방식은 고전 데이터의 인덱스 중 1이 위치한 인덱스와 같은 위치의 큐비트에 X 게이트 연산을 적용한다. 이 방식은 클래식 데이터와 양자 데이터 사이에 사용될 수 있는 방식으로 클래식 데이터를 양자 데이터로 변경하여 CNOT 게이트를 수행하는 연산을 대신 할 수 있도록 한다. 또한, 양자 회로 깊이를 줄이기 위해 Swap 게이트를 사용하지 않으며 Shift 연산은 배열의 인덱스를 변경하여 구조를 설계하였다.

[그림 2]은 G함수에 대한 양자회로 (1)큐비트 최적화 된 양자회로와 (2)깊이 최적화된 양자회로의 회로를 보여준다. 그림의 두 회로에서 입력 m은 사전 결정된 고전 데이터 이며 σ은 사전 결정된 양자 데이터 이다. 큐비트 $|a\rangle$~$|d\rangle$는 G의 입력을 나타낸다. $|a\rangle$~$|d\rangle$ 큐비트의 입력 순서는 다음과 같다:

$G(a,b,c,d) = G(v_0,v_4,v_8,v_{12})$
$G(a,b,c,d) = G(v_1,v_5,v_9,v_{13})$
$G(a,b,c,d) = G(v_2,v_6,v_{10},v_{14})$
$G(a,b,c,d) = G(v_3,v_7,v_{11},v_{15})$
$G(a,b,c,d) = G(v_0,v_5,v_{10},v_{15})$
$G(a,b,c,d) = G(v_1,v_6,v_{11},v_{12})$
$G(a,b,c,d) = G(v_2,v_7,v_8,v_{13})$
$G(a,b,c,d) = G(v_3,v_4,v_9,v_{14})$

(1) 큐비트 최적화 양자 회로는 역 연산을 통해 큐비트을 재사용함으로써, 큐비트의 수를 줄이는 대신 회로의 깊이를 증가시킨다. 이 방법은 사용된 64 큐비트 σ를 역 연산을 통해 재사용하여 모든 압축 함수가 단일 64 큐비트 σ로 작동하도록 한다. 이 회로에는 큐비트의 수를 줄이기 위한 두 개의 역연산 지점(inverse point 1, 2) 이 있으며, 두 지점에서 시그마 σ는 $|0\rangle$으로 재설정된다. 이러한 리셋을 통해 시그마 σ로 할당된 큐비트는 함수 내에서 뿐만 아니라 모든 라운드에서 계속 사용할 수 있다. [Algorithm 1]은 압축함수 G에 대한 큐비트 최적화 양자 회로의 의사 코드(pseudo-code)을 보여준다. 3, 19 라인은 역 지점(Reverse Points)이며, 6, 22 라인은 각 역 지점의 역연산 타이밍을 나타낸다. 2, 5, 12, 18, 21, 26번 줄에서는 두 가지 덧셈기, 즉 깊이 (6n−2) 덧셈기와 깊이 (2n+3) 덧셈기를 사용하여 ADD가 구현된다. 인덱스 연산들에 대해서는 시프트 대신 연산 인덱스 순서를 조정하여 깊이가 증가하지 않았으며, SWAP 게이트 대신 논리 배열을 사용하여 큐비트의 물리적 위치를 조정함으로써 깊이를 줄였다. 4, 20 라인의 Classic to Quantum 함수는 m과 σ가 큐비트 간의 Quantum-to-Quantum 연산이 아니라 고전적 상수 값의 상태에 따른 Classic-to-quantum 연산이 되도록 설계되었다. 이 접근 방식은 m과 시그마 업데이트에 사용되는 큐비트 및 양자 게이트의 수를 줄이는 데 기여한다. 상수 m은 이미 알려진 상수이므로, m은 사전 계산 테이블에 저장되고, 각 라운드에서 m의 인덱스 비트 값이 1인 부분과 동일한 시그마 인덱스에서 X 게이트가 작동된다. 이러한 작업은 CNOT 게이트보다 비용이 적게 드는 X 게이트를 사용하여 대체할 수 있으므로, 양자 자원의 측면에서 매우 효율적이다.

(2) 깊이 최적화 양자회로는 ancilla 큐비트의 사용을 통해 양자회로 깊이를 줄인다. 64 큐비트로 할당된 σ은 사용시 매번 할당하여 사용하고 역연산이 존재하지 않으므로 역연산 지점도 존재하지 않는다. σ에서 할당된 큐비트는 재사용 되지 않으며 모든 라운드의 함수 내에서 계속 할당된다. [Algorithm 2]는 압축함수 G에 대한 깊이 최적화 양자 회로의 의사 코드(pseudo-code)을 보여준다. 2, 4, 10, 16, 18, 22 라인에서 ADD는 깊이 (6n−2) 덧셈기와 깊이 (2n+3) 덧셈기를 사용하여 구현되며, SWAP 게이트 대신 논리 배열을 사용하여 큐비트의 물리적 위치를 조정함으로써 깊이를 줄였다.

3번과 17번 줄의 Classic to Quantum 함수는 m과 σ가 큐비트 간의 Quantum-to-Quantum 연산이 아니라 고전적 상수 값의 상태에 따른 Classic-to-Quantum 연산이 되도록 설계되었다. 이 방법은 역 연산을 포함하지 않으므로 전체 깊이를 줄일 수 있다. 큐비트 최적화 양자 회로와 마찬가지로, 이미 알려진 상수 m은 사전 계산 테이블에 저장되며, X 게이트는 각 라운드에서 m의 인덱스 비트 값이 1인 부분과 동일한 σ 인덱스에서 작동된다.

3.1.2. Argon2 양자자원 추정 결과

해당 논문[10]에서는 Argon2에 대한 두가지 관점의 양자회로를 제안하였으며 각 양자회로는 큐비트 감소 및 깊이 감소에 중점을 두고 있다. [표 1] 및 [표 2]는 각각 큐비트 및 깊이 최적화 된 Argon2에 양자회에 대해 그루버 알고리즘에 필요한 자원추정 결과를 보여준다. 두 양자회로에는 (2n+3)깊이의 simple 덧셈기가 사용되었다. 결과와 같이 두 양자회로의 양자자원 차이는 큐비트 12,740 개, 깊이 약 13,967,032 로 각각의 측면에서 최적화 된 결과를 보여준다.

Algorithm. 1. Qubit-optimized quantum circuit for the compression function(G)

1: $a \leftarrow ADD(b, a)$
2: @Reverse Point
3: $\sigma \leftarrow$ Classic to Quantum($m[\sigma[r][2*i+0]]$)
4: $a \leftarrow ADD(\sigma, a)$
5: #Reverse
6: for (k=0 to length(d))
7: $d[k] \leftarrow CNOT(a[k], d[k])$
8: for (k=0 to 64)

9: $d_{b1} \times$ append($d[(k+32) \bmod 64]$)
10: $d = d_{b1}$

11: $c \leftarrow ADD(d, c)$
12: for (k=0 to length(b)):
13: $b[k] \leftarrow CNOT(c[k], b[k])$

14: for (k=0 to 64):
15: $b_{b1} \times$ append($b[(k+24) \bmod 64]$)
16: $b = b_{b1}$

17: $a \leftarrow ADD(b, a)$

18: @Reverse Point
19: $\sigma \leftarrow$ Classic to Quantum($m[\sigma[r][2*i+1]]$)
20: $a \leftarrow ADD(\sigma, a)$
21: #Reverse

22: for (k=0 to 64):
23: $d_{b2} \times$ append($d[(k+16) \bmod 64]$)
24: $d = d_{b2}$

25: $c \leftarrow ADD(d, c)$
26: for (k=0 to 64):
27: $b_{b2} \times$ append($b[(k+64) \bmod 64]$)
28: $b = b_{b2}$

〔Algorithm 1〕 압축함수 G에 대한 큐비트 최적화 양자 회로

Algorithm. 2. Depth-optimized quantum circuit for the compression function(G)

1: $a \leftarrow ADD(b, a)$

2: $\sigma \leftarrow$ Classic to Quantum($m[\sigma[r][2*i+0]]$)
3: $a \leftarrow ADD(\sigma_1, a)$

4: for (k=0 to length(d))
5: $d[k] \leftarrow CNOT(a[k], d[k])$
6: for (k=0 to 64)
7: $d_{b1} \times$ append($d[(k+32) \bmod 64]$)
8: $d = d_{b1}$

9: $c \leftarrow ADD(d, c)$
10: for (k=0 to length(b)):
11: $b[k] \leftarrow CNOT(c[k], b[k])$

12: for (k=0 to 64):
13: $b_{b1} \times$ append($b[(k+24) \bmod 64]$)
14: $b = b_{b1}$

15: $a \leftarrow ADD(b, a)$

16: $\sigma_2 \leftarrow$ Classic to Quantum($m[\sigma[r][2*i+1]]$)
17: $a \leftarrow ADD(\sigma_2, a)$

18: for (k=0 to 64):
19: $d_{b2} \times$ append($d[(k+16) \bmod 64]$)
20: $d = d_{b2}$

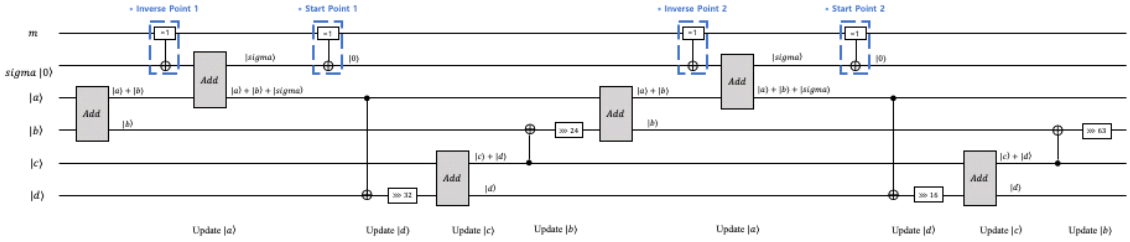
21: $c \leftarrow ADD(d, c)$

22: for (k=0 to 64):
23: $b_{b2} \times$ append($b[(k+64) \bmod 64]$)
24: $b = b_{b2}$

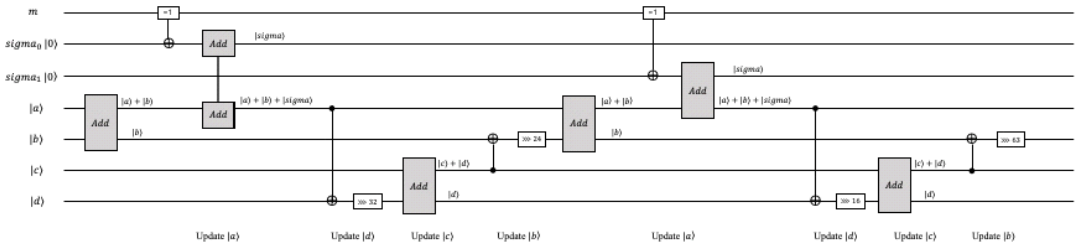
〔Algorithm 2〕 압축함수 G에 대한 깊이 최적화 양자 회로

3.2. Scrypt

Scrpyt는 2009년 Colin Percival이 제안한 password-based 키 도출 함수로, 계산 집약적이고 메모리 사용이 많은 작업을 하도록 설계되었으며, 특히 ASICs 및 GPUs(그래픽 처리 장치)를 사용하는 하드웨어 공격에 대해 자항성을 가진다. 또한, Scrypt는 비밀번호 해싱 및 암호화폐와 같은 높은 보안이 요구되는 애플리케이션에 적합하다. 〔Algorithm 3〕는 Scrpyt에 대한 전체 동작을 보여주며 〔Algorithm 4〕는 Scrypt 내부에서 사용되는 $BLOCKMIX_{sala8}$과 $SMix$ 연산을 나타낸다. salt는 무작위로 생성된 값이며 N은 CPU/메모리 비용. r은 블록 크기, p는 병렬 수를 나타내는 파라미터이다.



(1) G에 대한 큐비트 최적화 양자회로



(2) G에 대한 깊이 최적화 양자회로

〔그림 2〕 Agron2 G 함수 최적화 양자회로

〔표 1〕 Argon2에 대한 양자자원 추정 결과 (큐비트 최적화, Adder: simple)

Function	Qubit	1qCliford	CNOT	Toffoli	Full Depth
Initial	1090	(None)			
Update		(None)			
Final		1.03×2^{6}	1.98×2^{18}	1.66×2^{17}	1.25×2^{19}
blake2b		1.93×2^{10}	1.85×2^{23}	1.55×2^{22}	1.18×2^{24}
Total		1.99×2^{10}	1.91×2^{23}	1.6×2^{22}	1.21×2^{24}

〔표 2〕 Argon2에 대한 양자자원 추정 결과 (깊이 최적화, Adder: simple)

Function	Qubit	1qCliford	CNOT	Toffoli	Full Depth
Initial	13,830	(None)			
Update		(None)			
Final		1.12×2^{5}	1.98×2^{18}	1.66×2^{17}	1.56×2^{17}
blake2b		1.05×2^{10}	1.85×2^{23}	1.55×2^{22}	1.46×2^{22}
Total		1.08×2^{10}	1.91×2^{23}	1.6×2^{22}	1.51×2^{22}

Algorithm. 3. Scrypt function

input: password, passord length, salt, N, r, p

1 : $PBKDF2_{SHA256}(pwd, pwdlen, salt, saltlen, 1, B, p \times 128 \times r)$

2:for (i=0 to i<p):

3: $SMix(B_i, N)$ // $B_i \leftarrow MF(B_i, N)$

// $DK \leftarrow PBKDF2(p, B, 1, dkLen)$

4 : $PBKDF2_{SHA256}(pwd, pwdlen, B, p \times 128 \times r, 1, buf, buflen)$

〔Algorithm 3〕 Scrypt 알고리즘

Algorithm. 4. $SMix, BLOCKMIX_{salsa8}$ in Scrypt

1: function $SMix(B, N)$

2: $X \leftarrow B$

3: for (i=0 to i<N):

4: $V_i \leftarrow X$

5: $X \leftarrow BLOCKMIX_{salsa8}(X)$

6: for (i=0 to i<N):

7: $j \leftarrow integerify(X) \bmod N$

8: $X \leftarrow BLOCKMIX_{salsa8}(X \oplus V_j)$

9: function $BLOCKMIX_{salsa8}$

10: $X \leftarrow B_{2r-1}$

11: for (i=0 to $i < 2 \times r$):

12: $X \leftarrow Salsa_{20/8}(X)$

13: $Y_i \leftarrow X$

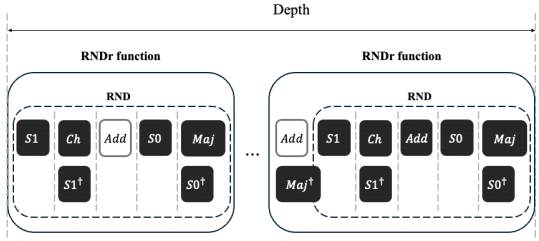
〔Algorithm 4〕 Scrypt 내부함수 알고리즘

3.2.1. Scrypt 양자회로

Scrypt의 주요 연산은 $SMix$와 $PBKDF2_{SHA256}$을 포함하며 $SMix$ 함수 내에서는 Salsa20/8 (Salsa20의 8라운드 버전)가 사용되고 $PBKDF2_{SHA256}$ 함수에서는 SHA-256이 사용된다. 해당 논문[11]에서 구현한 Scrypt 양자회로의 궁극적인 목표는 양자회로의 복잡성, 즉 시간 및 공간 복잡도의 곱(큐비트×깊이, DW-cost)을 줄이는 것이다. 이를 위해 그루버 알고리즘의 Oracle에서 사용될 Scrypt 양자회로의 큐비트 및 깊이 trade-off를 조정했다. 이러한 목표는 병렬구조 및 큐비트 최적화 기법 도입으로 달성됐다.

$PBKDF2_{SHA256}$ 함수의 SHA-256은 ancilla 큐비트 수를 많이 줄이면서 깊이를 조금 증가시키는 방식으로 구현되었다. SHA-256 연산은 수식(1)과 같다.

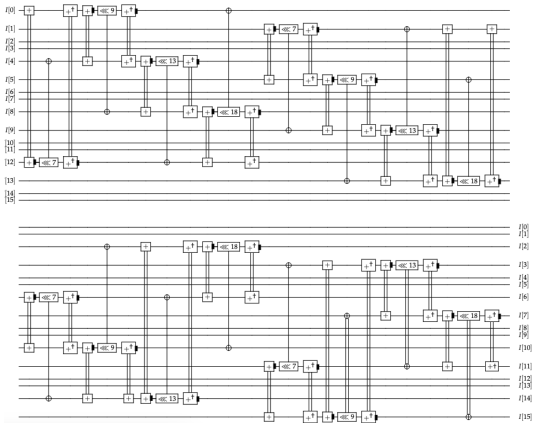
$$
\begin{aligned}
&Ch(x,y,z) = ((x \,\&\, y \hat{}\, z)) \hat{}\, z) \\
&Maj(x,y,z) = ((x \,\&\, (y|z))|(y \,\&\, z)) \\
&SHR(x,n) = (x \gg n) \\
&ROTR(x,n) = ((x \gg n)|(x \ll (32-n))) \qquad (1) \\
&S0(x) = (ROTR(x,2) \hat{}\, ROTR(x,13) \hat{}\, ROTR(x,22)) \\
&S1(x) = (ROTR(x,6) \hat{}\, ROTR(x,11) \hat{}\, ROTR(x,25)) \\
&s0(x) = (ROTR(x,7) \hat{}\, ROTR(x,18) \hat{}\, SHR(x,3)) \\
&s1(x) = (ROTR(x,17) \hat{}\, ROTR(x,19) \hat{}\, SHR(x,10))
\end{aligned}
$$



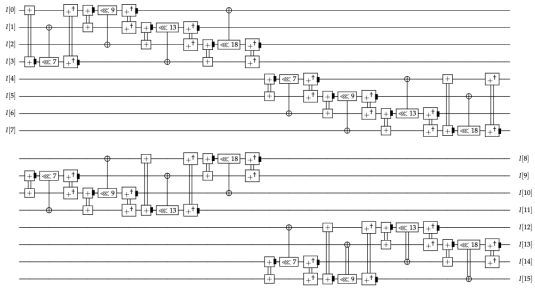
〔그림 3〕 $PBKDF2_{SHA256}$의 RNDr 함수

해당 SHA-256 양자 구현에서는 큐비트 재사용을 위해 수행되는 역연산 $S1^{\dagger}$, $S0^{\dagger}$, $Maj^{\dagger}$이 후속 연산들과 병렬로 설계되어 전체 깊이를 줄였다. [그림 3]과 같이 RND 연산 내에서 $S1^{\dagger}$, $S0^{\dagger}$는 후속 Ch, Maj 연산과 병렬로 실행되며 $Maj^{\dagger}$는 다음 RND_r 함수와 병렬로 동작한다. 이 방법을 통해 SHA-256 연산이 수행될 때 ancilla 큐비트를 8,128개 줄였으며 깊이는 약 6으로 아주 조금 증가하였다.

$SMix$ 함수 내에서의 Salsa20/8 (Salsa20의 8라운드 버전) 양자회로는 [그림 4] [그림 5]와 같다. 그림에서 $+$는 in-place 덧셈을 나타내며 $+^{\dagger}$는 역연산을 나타낸다. 각 CNOT 게이트는 control 큐비트를 오른쪽으로 n비트 시프트($\ll n$) 한 뒤 수행된다. I는 16×32 배열의 큐비트를 나타내며 $I[i][j]\,(0 \le i \le 16, 0 \le j \le 32)$로 표기된다. 따라서 $I[0], \cdots, I[15]$는 각각 32 큐비트 배열을 나타낸다.



〔그림 4〕 Salsa20/8의 열(column) 연산



〔그림 5〕 Salsa20/8의 행(row) 연산

3.2.2. Scrypt 양자자원 추정 결과

[11]에서는 Argon2에 대한 큐비트-깊이의 사이의 trade-off를 고려한 회로 복잡도 최적화 양자회로를 제안하였다. [표 3]은 Scrypt 내부 함수 SHA3, Salsa20/8에 대한 양자자원 추정 결과를 보여준다. 제안한 양자회로 최적화를 통해 SHA3 in *PBKDF*2 에서는 8,128개의 큐비트를 줄이고 깊이는 오직 6 만큼 증가하였으며 Salsa20/8 in *SMix* 내에서는 역연산을 통해 큐비트를 재사용하고 일부 병렬 연산을 통해 전체 깊이를 줄였다. 또한, m-XOR 및 index-rotation 기법을 통해 양자게이트를 줄였다. 결과적으로 Scrypt 양자회로에 대해 큐비트 수를 크게 줄이는것과 동시에 깊이는 거의 증가하지 않았다.

Ⅳ. 결 론

본 논문에서는 패스워드 크래킹 암호인 Argon2, Scrypt에 대한 양자회로 구현 동향을 살펴보았다. Argon2 양자회로는 큐비트 및 깊이에 최적화 된 양자회로를 각각 제시하였으며 Scrypt 양자회로는 큐비트 및 깊이의 trade-off를 가진 단일 양자회로를 제안했다. 각 양자회로들는 큐비트 및 깊이 측면에서 최적화를 진행하기 위해 여러 기법들을 도입하였으며 결과적으로 두 가지 측면에서의 최적화가 모두 중요하다는 것을 확인했다. 마지막으로 각 양자회로의 양자자원 추정을 통해 최적화 결과를 확인하였다.

참 고 문 헌

[1] O. M. Ogbanufe, C. Baham, "Using multi-factor authentication for online account security: Examining the influence of anticipated regret," Information Systems Frontiers, vol. 25, no. 2, pp. 897-916, May 2023.

[2] A. P. Umejiaku, P. Dhakal, V. S. Sheng, "Balancing password security and user convenience: Exploring the potential of prompt models for password generation," Electronics, vol. 12, no. 10, pp. 2159, May 2023.

[3] A. Biryukov, D. Dinu, D. Khovratovich, "Argon2: new generation of memory-hard functions for password hashing and other applications," in 2016 IEEE European Symposium on Security and

〔표 3〕 Scrypt에 대한 양자자원 추정 결과

Function	Qubit	Quantum Gates					T-depth	Full depth	DW-cost
		Toffoli	CNOT	X	T	$T^{\dagger}$			
SHA3 in *PBKDF*2	17,100	58,448	137,888	63,231	179,230	225,774	292,240	138,358	1.1×2^{31}
Salsa20/8 in *SMix*	1040	16,592	145,776	16,060	57,448	58,424	82,960	35,050	1.28×2^{26}

Privacy (EuroS&P), Mar. 2016, pp. 292-302.

[4] C. Percival, "Stronger key derivation via sequential memory-hard functions," 2009.

[5] Q. Liu, B. Preneel, Z. Zhao, M. Wang, "Improved quantum circuits for AES: reducing the depth and the number of qubits," in International Conference on the Theory and Application of Cryptology and Information Security, Singapore: Springer Nature Singapore, Dec. 2023, pp. 67-98.

[6] G. Song, K. Jang, H. Kim, W. K. Lee, Z. Hu, H. Seo, "Grover on SM3," in International Conference on Information Security and Cryptology, Cham: Springer International Publishing, Dec. 2021, pp. 421-433.

[7] M. Rahman, G. Paul, "Grover on KATAN: Quantum resource estimation," IEEE Transactions on Quantum Engineering, vol. 3, pp. 1-9, Dec. 2022.

[8] M. Grassl, B. Langenberg, M. Roetteler, R. Steinwandt, "Applying Grover's algorithm to AES: quantum resource estimates," in International Workshop on Post-Quantum Cryptography, Cham: Springer International Publishing, Feb. 2016, pp. 29-43.

[9] L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, Jul. 1996, pp. 212-219.

[10] G. Song, S. Eum, H. Kwon, M. Sim, M. Lee, H. Seo, "Optimized Quantum Circuit for Quantum Security Strength Analysis of Argon2," Electronics, vol. 12, no. 21, pp. 4485, Dec. 2023.

[11] G. Song, H. Seo, "Grover on Scrypt," Electronics, vol. 13, no. 16, pp. 3167, Aug. 2024.

〈저자소개〉

송 경 주 (Gyeong-Ju Song)

학생회원

2021년 2월 : 한성대학교 IT융합공학부 졸업

2023년 2월 : 한성대학교 IT융합공학부 석사

2023년 3월~현재 : 한성대학교 정보컴퓨터공학과 박사과정

<관심분야> 양자컴퓨팅, 암호구현, 정보보안

서 화 정 (Hwa-Jeong Seo)

종신회원

2010년 2월 : 부산대학교 컴퓨터공학과 졸업

2012년 2월 : 부산대학교 컴퓨터공학과 석사

2016년 2월 : 부산대학교 컴퓨터공학과 박사

2017년 4월~2023년 2월 : 한성대학교 IT융합공학부 조교수

2023년 3월~현재 : 한성대학교 융합보안학과 부교수

<관심분야> 정보보안, 암호구현