KIPS Trans. Comp. and Comm. Sys.
Vol.11, No.12 pp.453~460
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2022.11.12.453

Analysis of Grover Attack Cost and Post-Quantum Security Strength Evaluation for Lightweight Cipher SPARKLE SCHWAEMM

Yang Yu Jin[†] · Jang Kyung Bae[††] · Kim Hyun Ji[††] · Song Gyung Ju[†] · Lim Se Jin[†] · Seo Hwa Jeong[†††]

ABSTRACT

As high-performance quantum computers are expected to be developed, studies are being actively conducted to build a post-quantum security system that is safe from potential quantum computer attacks. When the Grover's algorithm, a representative quantum algorithm, is used to search for a secret key in a symmetric key cryptography, there may be a safety problem in that the security strength of the cipher is reduced to the square root. NIST presents the post-quantum security strength estimated based on the cost of the Grover's algorithm required for an attack of the cryptographic algorithm as a post-quantum security requirement for symmetric key cryptography. The estimated cost of Grover's algorithm for the attack of symmetric key cryptography is determined by the quantum circuit complexity of the corresponding encryption algorithm. In this paper, the quantum circuit of the SCHWAEMM algorithm, AEAD family of SPARKLE, which was a finalist in NIST's lightweight cryptography competition, is efficiently implemented, and the quantum cost to apply the Grover's algorithm is analyzed. At this time, the cost according to the CDKM ripple-carry adder and the unbounded Fan-Out adder is compared together. Finally, we evaluate the post-quantum security strength of the lightweight cryptography SPARKLE SCHWAEMM algorithm based on the analyzed cost and NIST's post-quantum security requirements. A quantum programming tool, ProjectQ, is used to implement the quantum circuit and analyze its cost.

Keywords : Quantum Computer, Lightweight Block Cipher, SPARKLE, Grover Search Algorithm

경량암호 SPARKLE SCHWAEMM에 대한 Grover 공격 비용 분석 및 양자 후 보안 강도 평가

양 유 진[†] · 장 경 배[††] · 김 현 지[††] · 송 경 주[†] · 임 세 진[†] · 서 화 정[†††]

요 약

고성능 양자 컴퓨터의 개발이 기대됨에 따라 잠재적인 양자 컴퓨터의 공격으로부터 안전한 양자 후 보안 시스템 구축을 위한 연구들이 활발하게 진행되고 있다. 대표적인 양자 알고리즘 중 하나인 Grover 알고리즘이 대칭키 암호의 키 검색에 사용될 경우, 암호의 보안 강도가 제곱근으로 감소되는 안전성의 문제가 발생할 수 있다. NIST는 암호 알고리즘의 공격에 필요로 하는 Grover 알고리즘의 비용을 기준으로 추정한 양자 후 보안 강도를 대칭키 암호에 대한 양자 후 보안 요구사항으로 제시하고 있다. 대칭키 암호의 공격에 대한 Grover 알고리즘의 추정 비용은 해당하는 암호화 알고리즘의 양자 회로 복잡도에 의해 결정된다. 본 논문에서는 NIST의 경량암호 공모전 최종 후보에 오른 SPARKLE의 AEAD군인 SCHWAEMM 알고리즘의 양자 회로를 효율적으로 구현하고, Grover 알고리즘을 적용하기 위한 양자 비용에 대해 분석한다. 이때, 암호화 순열 과정 중에 사용되는 덧셈기와 관련하여 CDKM ripple-carry 덧셈기와 Unbounded Fan-Out 덧셈기에 따른 비용을 같이 비교한다. 마지막으로, 분석한 비용과 NIST의 양자 후 보안 요구사항을 기반으로 경량암호 SPARKLE SCHWAEMM 알고리즘에 대한 양자 후 보안 강도를 평가한다. 양자 회로 구현 및 비용 분석에는 양자 프로그래밍 툴인 ProjectQ가 사용되었다.

키워드 : 양자 컴퓨터, 경량 블록 암호, SPARKLE, Grover Search Algorithm

※ This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (〈Q|Crypton〉, No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity, 100%).
※ 이 논문은 2022년 한국정보처리학회 ASK 2022의 우수논문으로 "경량암호 Sparkle에 대한 Grover 공격 비용 분석 및 양자 후 보안 강도 평가"의 제목으로 발표된 논문을 확장한 것임.

† 준 회 원 : 한성대학교 IT융합공학과 석사과정
†† 준 회 원 : 한성대학교 정보컴퓨터공학과 박사과정
††† 종신회원 : 한성대학교 IT융합공학부 조교수
Manuscript Received : August 30, 2022
Accepted : October 11, 2022
* Corresponding Author : Seo Hwa Jeong(hwajeong84@gmail.com)

1. 서 론

IBM과 Google을 선두로 하는 국제 IT 대기업들이 고성능 양자 컴퓨터의 개발을 앞당기고 있다. 양자 컴퓨터는 양자 역학의 특성을 이용하여 특정 문제를 효율적으로 모델링하고 해결할 수 있다. 이는 현재의 암호 시스템들이 기반하고 있는 소인수 분해, 이산대수 문제와 같은 수학적 난제 또한 빠르게 해결할 수 있을 것으로 예상된다. 이에 암호 학계에서는 잠재적인 양자 컴퓨터의 공격으로부터 안전한 양자 후 보안 시스템을 구축하기 위한 움직임을 보이고 있다.

공개키 암호의 경우, 가장 널리 사용되고 있는 ECC와 RSA가 양자 Shor 알고리즘에 의해 다항 시간 안에 해킹될 수 있음이 밝혀짐에 따라 새로운 암호가 필요한 상황이다. 이러한 필요성에 의해 NIST는 양자내성암호 표준화 공모전을 주최하였으며, 현재 1개의 공개키 암호/키 설정 알고리즘과 3개의 디지털 서명 알고리즘이 표준으로 선정되었고[1] 4개의 공개키 암호/키 설정 알고리즘 후보 알고리즘이 4 라운드에 진출한 상황이다[2].

대칭키 암호의 경우, 대표적인 양자 알고리즘인 Grover 알고리즘에 의해 비밀 키 전수 조사 복잡도를 $\sqrt{\ }$만큼 감소시킬 수 있다[3]. 2016년, NIST는 양자 후 암호 시스템들에 대한 양자 후 보안 요구 사항으로 AES 알고리즘 공격에 사용되는 Grover 알고리즘의 비용을 제시하였다[4]. 이에 따라 AES를 양자 회로로 최적화하여 구현하는 연구들을 시작으로[5, 6], 다양한 블록 암호들에 대한 Grover 공격 비용을 추정하는 연구들이 제시되고 있다[7-10]. 대칭키 암호에 대한 Grover 키 검색은 공격 대상 암호화 양자 회로를 얼마나 효율적으로 구현하는지에 따라 최종 비용이 결정되기 때문에 양자 회로를 최적화 구현하는 것이 상당히 중요하다.

본 논문에서는 NIST LWC(Lightweight Cryptography) 공모전의 후보 알고리즘인 SPARKLE[11]에 대한 Grover 양자 공격 비용을 분석한다. SPARKLE 순열 함수에는 ARX (Addition-Rotation-XOR) 구조의 연산이 사용되는데, 여기에 사용되는 양자 덧셈 연산을 위해 CDKM ripple-carry 덧셈기와 Unbounded Fan-Out 덧셈기를 구현하여 그 결과를 비교한다. 또한, 패딩 연산에 대한 큐비트 최적화와 선형 함수에 대한 큐비트 최적화를 통해 효율적인 SPARKLE 양자 회로를 제시한다. 최종적으로 추정한 비용을 기반으로, NIST가 제시한 보안 요구사항에 따라 덧셈기별 SPARKLE의 양자 후 보안 강도를 평가한다. NIST LWC 공모전의 최종 후보 알고리즘 중 하나인 SPARKLE에 대해 양자 후 보안성 평가를 진행함으로써 안전한 IoT 시스템을 구축하는 데 기여하고자 한다.

2. 관련 연구

2.1 Grover 검색 알고리즘

Grover 알고리즘은 1996년 Lov Grover에 의해 제안된 양자 검색 알고리즘으로, 양자 역학의 중첩(superposition)과 얽힘(entanglement) 원리를 이용하여 원하는 답을 도출해낼 수 있다[3, 5, 12]. 함수의 도메인 크기가 N 일 때, Grover 알고리즘을 적용하면 계산 복잡도가 $O(N)$에서 $\sqrt{N}$으로 줄어든다.

Grover 검색 알고리즘을 사용한 양자 키 검색은 크게 3가지 단계로 구성되며 알려진 평문-암호문 쌍에 대한 정답 키를 찾는다. 첫 번째 단계는 Initialization이다. 해당 단계에서는 검색 대상이 되는 n-큐비트 키에 Hadamard 게이트들을 적용시킨다. 이를 통해 n-큐비트 키는 중첩 상태로 존재하게 되고 2^n개의 키 값이 확률로서 존재하게 된다. Initialization에서는 알려진 평문을 큐비트로 준비하는 과정이 필요하며 이는 X 게이트들을 사용하여 구현된다. 두 번째 단계는 Grover 오라클(Oracle)이다. Grover 오라클에는 공격 대상 암호화 회로가 양자회로로 구현되어 자리한다. 암호화 양자 회로는 중첩 상태의 키로 알려진 평문을 암호화하며, 이를 통해 모든 키 값으로 암호화한 암호문이 중첩 상태로 존재하게 된다. 이후 중첩 상태의 암호문에서 알려진 암호문과 일치하는 경우 조건부 Z 게이트가 작동하여 정답인 키 값의 부호를 반전시켜준다. 마지막 단계인 확산 연산자(Diffusion operator)에서는 오라클에서 반환한 정답 키의 amplitude를 증폭시킨다. 확산 연산자는 오버헤드가 매우 적으며, 설계가 일반적이기 때문에 공격 비용 추정시 무시할 수 있다. Grover 검색 알고리즘은 오라클과 확산 연산자의 반복을 통해, 정답 키의 관측 확률을 충분히 높인 뒤, n-큐비트 키를 관측하여 높은 확률로 정답 키를 복구할 수 있다.

예를 들어 $N=4$ 일 때, Initialization 단계를 거치면 입력 큐비트들이 중첩상태가 된다. 00, 01, 10, 11 중 답이 10일 경우 Grover 오라클에서는 10에 해당하는 상태의 부호를 반전시켜준다. 부호 반전 이후 확산 연산자를 거치면 모든 검색 대상 상태의 amplitude가 증폭된다. 이후, 이 과정을 $\sqrt{4}$번, 즉 2번 반복하면 정답이 아닌 상태의 amplitude는 감소하고 정답인 10의 amplitude는 증폭되어 높은 확률로 정답을 관측할 수 있다.

2.2 양자 게이트

양자 컴퓨터 환경에서는 기존의 컴퓨터에서 사용되었던 AND, OR, XOR과 같은 논리 게이트를 사용할 수가 없기 때문에 이를 대체할 수단으로 양자 게이트를 사용한다. 양자게이트의 대표격으로는 X 게이트, CNOT 게이트, Toffoli 게이트, SWAP 게이트가 있다. X 게이트는 입력으로 들어온 단일 큐비트를 반전시켜주며 논리 게이트 중 NOT 게이트와 매칭될 수 있다. CNOT 게이트는 제어 큐비트(control qubit)가 1일 경우에만 타겟 큐비트(target qubit)를 반전시켜주는 게이트로 두 입력 간의 XOR 연산을 수행할 때 자주 사용된다. CCNOT이라고도 불리는 Toffoli 게이트는 제어 큐비트가 2개 있는 CNOT 게이트로 2개의 제어 큐비트가 모두 1이어야만 타겟 큐비트에 영향을 끼칠 수 있다. Toffoli 게이트가 4가지 기본 양자 게이트 중에 양자 비용이 가장 비싸다. SWAP 게이트는 입력으로 들어온 2개의 큐비트의 상태를 서로 변경시켜준다. 앞선 게이트들과 달리 SWAP 게이트는 양자 자원에 영향을 끼치지 않는다.

2.3 SPARKLE SCHWAEMM

Sponge 구조에 기반하여 설계된 SPARKLE은 NIST LWC 표준화 최종후보에 오른 경량 암호 알고리즘이다. SPARKLE 암호의 가장 핵심이 되는 연산인 SPARKLE Permutation은 ARX 연산 기반의 64-bit S- box인 Alzette와 파이스텔 구조의 선형 레이어로 구성된 diffusion layer를 사용하여 암호화 순열 작업을 수행한다.

SPARKLE은 AEAD군에 속하는 SCHWAEMM과 Hash 함수군에 속하는 ESCH로 구성되어 있다. 그 중 SCHWAEMM은 상태 초기화(Initialization)와 연관데이터 처리(Processing of associated data), 암호화(Encrypting), 완료(Finalization)까지 총 네 단계의 연산을 거친다. SCHWAEMM의 파라미터는 Key와 Nonce의 길이에 따라 128/128, 128/256, 192/192, 256/256이 있다. 4가지 파라미터 모두 구조가 같기 때문에 연산 단계를 설명할 때 128/128을 기준으로 두고 설명할 것이다.

상태 초기화 이전에 입력받은 데이터를 128-bit로 맞춰주기 위해서 입력 데이터 D에 대해 $D\|00000001\|0^i$ 형식이 되도록 데이터를 패딩한다. D에 0x01을 붙인 다음 128-bit가 될 때까지 0으로 빈 공간을 채워 넣는 방식이다.

상태 초기화 단계에선 128-bit의 Nonce와 Key를 차례로 연접하여 생성되는 내부 상태 S를 $SPARKLE256_{10}$에 넣어 초기화 해준다.

연관 데이터 처리 단계에서는 128-bit 크기로 패딩 처리된 연관데이터 A를 피드백 함수 $\rho 1$에 넣어 $(S,D)\mapsto$FeistelSwap$(S)\oplus D$ 연산을 수행하고, SPARKLE 암호화 순열과 XOR 연산을 거치며 S를 갱신해준다. 이때, A가 마지막 블록이 아니면 7라운드의 암호화 순열을 의미하는 $SPARKLE256_7$을 거치고, 마지막 블록이면 내부 상태의 오른쪽을 의미하는 S_R에 상수 $const_M$을 XOR 연산한 후 $SPARKLE256_{10}$을 거친다.

메시지 M을 암호화해주는 암호화 단계에서는 내부상태의 가장 왼쪽부터 t만큼만 반환하는 $trunc_t$함수와 두 입력값 S와 M을 XOR 연산해주는 $\rho 2$ 함수를 거쳐 암호문 C 가 생성된다. 암호문 또한 연관데이터와 마찬가지로 마지막 블록이 아닌 경우 $SPARKLE256_7$을 수행하고, 마지막 블록일 경우 S_R에 상수 $const_M$을 XOR한 후 $SPARKLE256_{10}$을 수행한다.

마지막으로 완료 단계에서는 $SPARKLE256_{10}$을 수행하여 S를 업데이트하고 키 K와 S_R을 XOR 연산하여 TAG를 생성한다. 여기에 이전 단계에서 생성한 암호문 C와 TAG 를 합친 값이 SCHWAEMM-128/128의 최종 결과값이다.

Fig. 1은 SCHWAEMM-128/128의 전체 흐름을 그림으로 나타낸 것이다.

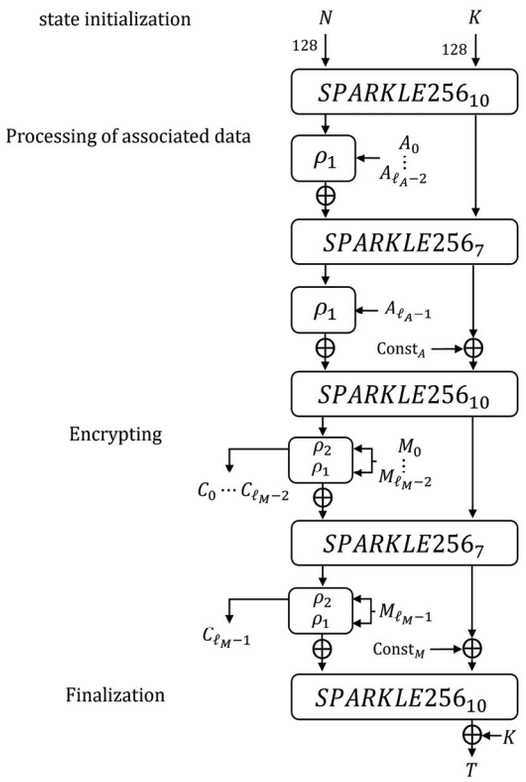

Fig. 1. SCHWAEMM-128/128 Flowchart

2.4 CDKM carry-ripple 덧셈기

CDKM(Cuccaro-DraperKutin-Moulton) carry-ripple 덧셈기는 ripple-carry 덧셈기를 향상시킨 버전으로, ripple-carry 덧셈기에서 사용되던 양자 게이트의 수와 회로 depth를 줄였다는 점에서 큰 장점을 갖는다[13]. 또, CDKM은 UMA 게이트를 구현할 때, 단순한 구조의 2-CNOT version을 사용한 ripple-carry 덧셈기와 달리 비교적 복잡하지만 병렬 연산이 가능한 구조의 3-CNOT version을 사용하여 회로의 depth를 줄였다는 특성을 갖는다. 이러한 CDKM은 $n \geq 4$인 회로에서만 유효하다.

2.5 Carry Look-ahead 양자 덧셈기

CLA(Carry Look-ahead adder)의 핵심은 캐리의 상태를 k, g, p 3가지로 나눠 계산하는 것이다. 캐리가 아예 발생하지 않는 상태를 k(killed), 무조건 캐리가 발생하는 상태를 g(generated), 이전 연산에서 넘어온 캐리 값을 알지 못해 캐리 발생 여부를 결정할 수 없는 상태를 p(propagated)라고 정의한다.

Draper이 제안한 Carry Look-ahead 양자 덧셈기는 CLA를 양자 회로로 구현한 것으로 모든 단계에서 두 숫자의 합과 캐리값이 이전 단계의 결과에 의존하지 않기 때문에 병렬 연산을 진행함으로써 양자 회로의 깊이를 줄일 수 있다. 캐리상태를 이용해서 양자 회로를 구성할 때 P, G, C, P^{-1} 총 4가지 유형이 $\lfloor \log n \rfloor$ round 씩 순차적으로 동작한다. P-round는 보조 공간에 $p[i,j]$를 계산해서 넣어주고, G-round는 모든 $g[i,j]$를 찾아 저장하며, C-round는 $G[i]=g[0,i]=c_i$를 수행한다. 마지막으로 P^{-1}-round에서는 P-round의 inverse 연산을 진행하며 P-round에서 저장된 값들을 비워준다. 양자 회로 구현 방법은 각 bit의 합 저장

위치에 따라 in-place와 out-of-place 방법으로 나뉜다. in-place는 각 bit의 합이 입력 bit에 저장되는 반면, out-of-place는 추가로 할당한 보조 큐비트(ancillary qubit)에 값을 저장한다. 전자는 depth보다 큐비트에 초점을 맞춘 것이고, 후자는 큐비트보다 depth에 집중한 방식이다[14, 15].

2.6 Unbounded Fan-Out 덧셈기

앞서 설명한 CDKM 덧셈기의 경우 n-bit의 모듈러 덧셈을 수행하는데 있어 1개의 보조 큐비트를 필요로 한다. 반면에 UFO(Unbounded Fan-Out) 덧셈기는 n-bit의 양자 덧셈을 수행하는데 있어 보조 큐비트를 전혀 사용하지 않는다는 것이다. 따라서 최소 큐비트 수인 2개만을 사용한 덧셈이 가능하다. 해당 양자 덧셈기는 CDKM 덧셈기보다 낮은 depth를 제공하며 최소한의 큐비트만을 사용하기 때문에 성능 또한 우수한 편이다[16].

3. 제안 기법

SPARKLE의 레퍼런스 C코드와 테스트 벡터를 사용하여 구현을 진행하였다. 양자 회로 구현 및 비용 분석에는 IBM에서 제공하는 양자 프로그래밍 툴인 ProjectQ가 사용되었다. 사용된 메시지 M과 연관데이터 A는 32-bit로 크기를 고정하였고 실험에 사용된 테스트 벡터는 Table 1과 같다.

SPARKLE 암호 순열 과정 중 Alzette 단계에서 덧셈기가 쓰인다. 본 제안에서는 CDKM, UFO 두 가지 방식의 덧셈기를 선정해 구현하였다.

알고리즘 1은 CDKM 양자 덧셈기에 대한 양자 회로 설계이다. x와 y는 덧셈기에 들어가는 입력값이고 c는 덧셈 연산 중에 사용되는 1 큐비트의 보조 큐비트이다. 양자 덧셈 결과인 s는 y에 저장되며 y 가 출력값으로 나온다.

알고리즘 2는 UFO 양자 덧셈기에 대한 양자 회로 설계이다. 알고리즘1과 마찬가지로 x와 y가 입력값으로 들어가고 y 에 덧셈 연산 결과 s가 저장되어 반환된다. 덧셈 연산시 관련 연구에서 언급했듯이 보조 큐비트가 전혀 사용되지 않기 때문에 CDKM 덧셈기보다 큐비트 수를 줄일 수 있다.

SPARKLE 순열 함수에는 라운드 상수 덧셈, ARX 기반의 Alzette 함수, 그리고 선형 함수가 사용된다. 라운드 상수 덧셈의 경우 사전에 알려진 라운드 상수가 사용되기 때문에 X 게이트만을 사용하여 간단히 구현할 수 있다. 순열 함수의 라운드마다 4개의 Alzette 함수가 작동하는데 이는 양자 회로로 구현 시, 병렬로 설계할 수 있다. CDKM 덧셈기를 사용하는 경우, 1개의 보조 큐비트가 필요한데, 본 구현에서는 병렬 덧셈을 위해 4개의 보조 큐비트를 할당한다. 물론 3개의 추가적인 보조 큐비트를 사용하지만 이를 통해 4개의 Alzette 함수가 병렬로 동작하여 회로 depth를 크게 줄일 수 있다. 선형 함수의 경우, XOR과 로테이션 연산만이 사용되기 때문에 CNOT 게이트와 Swap 연산을 통해 양자 회로로 구현할 수 있다.

Table 1. Test Vectors for Experiment

M / A	0x03020100
K / N	**128-bit** 0x0F0E0D0C0B0A09080706050403020100
	192-bit 0x17161514131211100F0E0D0C0B0A090807060 50403020100
	256-bit 0x1F1E1D1C1B1A1918171615141312111 00F0 E0D0C0B0A09080706050403020100

Algorithm 1 : Quantum circuit implementation of CDKM adder
Input: x[j], y[j], carry qubit c, n ≥ 4 (0≤j≤n-1)
Output: y

```
for j = 0 to n-2
    CNOT (x[j+1], y[j+1])
CNOT (x[1], c)
Toffoli (x[0], y[0], c)
CNOT (x[2], x[1])
Toffoli (c, y[1], x[1])
CNOT (x[3], x[2])
for j = 0 to n-5
    Toffoli (x[j+1], y[j+2], x[j+2)
    CNOT (x[j+4], x[j+3])
Toffoli (x[n-4], y[n-3], x[n-3])
CNOT (x[n-2], y[n-1])
CNOT (x[n-1], y[n-1])
Toffoli (x[n-3], y[n-2], y[n-1])
for j = 0 to n-3
    X y[j+1]
CNOT (c, y[1])
for j = 0 to n-3
    CNOT (x[j+1], y[j+2])
Toffoli (x[n-4], y[n-3], x[n-3])
for j = 0 to n-5
    Toffoli (x[i-5-i], y[n-4-i], x[n-4-i)
    CNOT (x[n-2-i], x[n-3-i])
    X y[n-3-i]
Toffoli (c, y[1], x[1])
CNOT (x[3], x[2])
X y[2]
Toffoli (x[0], y[0], c)
CNOT (x[2], x[1])
X y[1]
CNOT (x[1], c)
for j = 0 to n-1
    CNOT (x[j], y[j])
return y
```

Algorithm 1. Quantum Circuit Implementation of CDKM Adder

알고리즘 3은 SPARKLE 순열 함수에 대한 양자 회로 구현을 나타낸다.

알고리즘 4부터 7까지는 SCHWAEMM을 대표하여 SCHWAEMM-128/128의 알고리즘을 정리한 것이다.

Algorithm 2 : Quantum circuit implementation of UFO adder
Input: x[j], y[j], n (0≤j<n)
Output: y

```
1: for j = 1 to n-1
2:     CNOT (x[j], y[j])
3: for j = n-2 to 1 step -1
4:     CNOT (x[j], x[j+1])
5: for j = 0 to n-2
6:     Toffoli (y[j], x[j], x[j+1])
7: for j = n-1 to 1 step -1
8:     CNOT (x[j], y[j])
9:     Toffoli (y[j-1], x[j-1], x[j])
10: for j = 1 to n-2
11:     CNOT (x[j], x[j+1])
12: for j = 0 to n-1
13:     CNOT (x[j], y[j])
14 : return y
```

Algorithm 2. Quantum Circuit Implementation of UFO Adder

Algorithm 3 : Quantum circuit implementation of SPARKLE Permutation (SPARKLE256$_r$)
Input: x_{0-3}, y_{0-3}, ancillary qubits ac_{0-3}, Constant c_{0-7}
Output: x, y

```
// Round constant addition
1: y0 ← Xgates(y0, c(i%8))
2: y0 ← Xgates(y0, c(i%8))
// Alzette functions
3: (x0, y0) ← Alzette(x0, yo, c0, ac0)
4: (x1, y1) ← Alzette(x1, y1, c1, ac1)
5: (x2, y2) ← Alzette(x2, y2, c2, ac2)
6: (x3, y3) ← Alzette(x3, y3, c3, ac3)
// Linear layer function with only CNOT gates
7: (x0-3, y0-3) ← Linear layer(x0-3, y0-3)
8: return x, y
```

Algorithm 3. Quantum Circuit Implementation of SPARKLE Permutation

알고리즘 4는 SCHWAEMM의 상태 초기화 단계를 양자 회로로 구현한 것이다. 2-3번째 라인을 보면 내부 상태S를 초기화하기 위하여 CNOT 게이트를 사용하여 S에 각각 오른쪽부터 N과 K를 넣었다. 그 후, 10라운드 도는 SPARKLE256을 호출하여 알고리즘 3에 언급된 연산을 수행한다.

알고리즘 5는 연관데이터 처리 단계의 양자 회로 구현으로, 1번째 라인은 S_R과 사전에 정의된 상수 $const_A$를 XOR 연산하는 부분이다. 원칙상 $const_A$에 큐비트를 할당해야 하지만 상수가 이곳에서 사용된 후 다시 쓰이지 않고, $0 \oplus (1 \ll 2) = 4 = (0100)_2$로 사용되는 값이 정해져 있기 때문에 새로운 큐비트를 할당하지 않고 S_R의 해당하는 위치에 바로 X 게이트를 적용하였다. 이를 통하여 CNOT 게이트 4개의 낭비를 막았다. 또한, $const_A$를 할당하지 않음으로써 4 큐비트 절약하였다.

Algorithm 4 : Quantum circuit implementation of State_Init
Input: carry-qubit c, K[j], N[j], S_L[j+128], S_R[j] (0≤j<128)
Output: $S = S_L \| S_R$

```
1: for j = 0 to 127
2:     CNOT (K[j], SR[j])
3:     CNOT (N[j], SL[j])
4: SPARKLE256(S, c, 10)
5: return S = SL||SR
```

Algorithm 4. Quantum Circuit Implementation of State Initialization

Algorithm 5 : Quantum circuit implementation of Processing_AD
Input: carry-qubit c, S_R, S_x[j], S_y[j+32], A (159<j<192)
Output: $S = S_L \| S_R$

```
1: X SR[26]
2: for j = 0 to 31
3:     SWAP (Sx[j+64], Sx[j])
4:     SWAP (Sy[j+64], Sy[j])
5: for j = 0 to 31
6:     CNOT (Sx[j+64], Sx[j])
7:     CNOT (Sy[j+64], Sy[j])
8: for j = 0 to 31
9:     CNOT (A[j], Sx[j+64])
10:    CNOT (A[j+32], Sy[j+64])
11: SPARKLE256(S, c, 10)
12: return S = SL||SR
```

Algorithm 5. Quantum Circuit Implementation of Processing of Associated Data

이후, 3-10번째 라인은 $\rho 1$ 함수를 구현한 부분이다, 이 함수를 적용하기 위해서는 M과 XOR 연산하기에 앞서 FeistelSwap$(S) = S_2 \| (S_2 \oplus S_1)$ 연산을 먼저 수행해줘야 한다. SWAP 게이트를 사용하여 S_{L1}과 S_{L2}의 값을 교환하고 CNOT 게이트를 통해 S_L의 각각 왼쪽, 오른쪽 값을 의미하는 S_x와 S_y에 XOR 연산을 하면 FeistelSwap(S) 단계를 거친 S가 나온다. 이를 M과의 XOR 연산까지 마치면 $\rho 1$ 함수가 끝난다. 이후, $SPARKLE256_{10}$에 S를 넣어 SPARKLE 순열 라운드를 10번 수행하면 연관데이터 처리 단계가 끝이 난다. S_L과 S_R이 연결된 상태 S가 해당 과정의 반환값이다.

Algorithm 6 : Quantum circuit implementation of Encrypting
Input: M, M_len=32, S_L[j], C (223<j<256)
Output: S_R, C

```
1: for j = 0 to M_len
2:     CNOT (M[j], C[j])
3: for j = 255 to 255-1-M_len
4:     CNOT (SL[j], C[j])
5: X SL[26]
6: X SL[25]
7: return SR, C
```

Algorithm 6. Quantum Circuit Implementation of Encrypting

Algorithm 7 : Quantum circuit implementation of Finalization
Input: carry-qubit c, K[j], S[j+128], M, C (0≤j<128)
Output: C ‖ S_R

1: ρ **1**(S_L, M)
2: **SPARKLE256**(S, c, 10)
3: **for** j = 0 **to** 128-1
4: **CNOT** (K[j], S_R[j])
5: **return** C ‖ S_R

Algorithm 7. Quantum Circuit Implementation of Finalization

알고리즘 6은 메시지를 암호화하는 Encrypting 과정이다. 1-2번째 라인에서는 초기화된 32-큐비트 암호문 C에 CNOT 게이트를 사용하여 M을 넣어준다. 3-4번째 라인에서 $\rho 2$, $trunc_t$ 함수를 진행하는데, $trunc_t$를 거치면 t만큼만 사용되기 때문에 두 입력값을 단순 XOR하는 $\rho 2$ 함수를 모든 입력이 아닌 메시지의 길이만큼만 수행하였다. 이는 $\rho 2$ 함수와 $trunc_t$함수를 동시에 실행하여 메시지 블록의 크기가 128보다 작을 경우 덜 수행할 수 있도록 한 것이다. 그 결과, 불필요한 연산을 줄임으로써 CNOT 게이트의 낭비를 막을 수 있다.

5-6번째 라인은 알고리즘4처럼 상수 $const_M$과 S_R를 XOR 연산하는 과정이다. $const_M$이 $2 \oplus (1 \ll 2) = 6 = (0110)_2$이기 때문에 S_R의 해당하는 위치에 X 게이트를 적용하였고, 이를 통해 게이트 비용과 4 큐비트를 절약하였다. Encrypting을 마치고 나면 암호문 C와 내부 상태 S_R이 결과값으로 반환된다.

알고리즘 7은 알고리즘 1~3을 거쳐 나온 값들을 종합해준다. 1-2번째 라인에서 알고리즘 3의 $\rho 1$과 $SPARKLE256_{10}$을 차례로 수행할 때 A 대신 M이 입력으로 들어간다. TAG를 생성하는 3-4번째 라인에서는 CNOT 게이트를 사용하여 S_R에 K와 S_R를 XOR 연산한 값을 저장한다. 그 결과 S_R에 저장된 값이 TAG가 된다. 5번째 라인은 최종 값을 반환해주는 부분으로 최종 반환 값은 암호문 C와 S_R 즉, TAG가 연접된 형태이다. 이후 Finalization을 거쳐 나온 최종 반환 값 $C \| TAG$을 관측하면 연산에 사용된 자원을 확인할 수 있다.

4. 평 가

본 논문에서는 양자 회로 구현과 테스트 벡터 검증, 사용된 양자 자원 추정을 위해 IBM의 양자 프로그래밍 툴인 ProjectQ가 사용되었다. 양자 회로 시뮬레이션을 위해 ProjectQ의 내부 라이브러리인 ClassicalSimulator를 사용하여, 구현한 양자 회로에 대한 암호화 테스트 벡터를 확인하였다. 또 다른 내부 라이브러리인 ResourceCounter는 구현에 사용된 양자 게이트들과 큐비트, 회로 depth를 추정하는데 사용되었다.

Table 2는 SCHWAEMM에 대한 모든 파라미터에 대한 양자 회로 구현 비용들을 나타내며 양자 CDKM 덧셈기가 사용되었다.

Table 3은 SCHWAEMM에 대한 양자 회로 구현 비용을 나타내며 양자 UFO 덧셈기가 사용되었다.

Table 2. Quantum Resources for Quantum Circuits of SCHWAEMM (CDKM Adder)

SCHWAEMM	qubits	CNOT gates	lqClifford gates	T gates	Depth
128/128	612	278,656	94,511	204,960	59,687
256/128	870	460,744	156,497	338,184	65,783
192/192	870	460,680	156,497	338,184	65,783
256/256	1,128	670,688	227,606	491,904	71,906

Table 3. Quantum Resources for Quantum Circuits of SCHWAEMM (UFO Adder)

SCHWAEMM	qubits	CNOT gates	lqClifford gates	T gates	Depth
128/128	608	252,256	66,631	208,320	53,254
256/128	864	417,184	112,145	343,728	58,711
192/192	864	417,120	112,145	343,728	58,711
256/256	1,120	607,328	163,094	499,968	64,192

Grover 알고리즘은 오라클과 확산 연산자의 반복으로 구성되어 있으나, 확산 연산자의 경우 오버헤드가 무시할 수 있을 정도로 적기 때문에 일반적으로 비용 추정을 할 때 무시된다. 따라서 Grover 알고리즘에 대한 공격 비용은 오라클에 위치하는 암호화 양자 회로를 얼마나 효율적으로 구현하는지가 관건이다. 오라클은 암호화에 1번, 리버스 연산에 1번해서 SPARKLE 양자 회로가 2번 순차적으로 동작한다. 따라서 오라클 한 번에 대한 비용은 Table 3 × 2(큐비트 제외)로 계산된다. 128-bit key를 사용하는 SPARKLE에 대한 Grover 공격에는 오라클이 약 $\sqrt{2^{128}}$번 즉, 2^{64}번 반복되기에 최종 비용은 Table 3 × 2^{64} × 2로 추정하며 Table 4에 나와 있다. Grover 양자 공격의 경우, 성능이 더 우수한 UFO 덧셈기를 사용한 구현에 대한 비용만을 추정하였다.

SPARKLE에 대한 Grover 공격 비용과 NIST의 양자 후보안 강도 기준을 비교하면, 128-bit 키를 사용하는 경우 가

Table 4. Cost of Grover Key Search for SCHWAEMM-128/128 (UFO Adder)

SCHWAEMM	qubits	Total gates	Total depth	Cost	NIST Secuirty
128/128	609	1.58×2^{83}	1.28×2^{79}	1.01×2^{163}	Not achieved (Level-1: 2^{170})
256/128	865	1.31×2^{84}	1.41×2^{79}	1.84×2^{163}	Not achieved (Level-1: 2^{170})
192/192	865	1.31×2^{116}	1.41×2^{111}	1.84×2^{227}	Not achieved (Level-2: 2^{233})
256/256	1,121	1.90×2^{148}	1.54×2^{143}	1.46×2^{292}	Not achieved (Level-3: 2^{298})

장 낮은 보안 강도인 Level-1을 달성하지 못한다. 192-bit, 256-bit 키를 사용하는 파라미터 또한 한 단계 낮은 보안 강도를 달성한다. 하지만 NIST가 인용하고 있는 공격 비용이 2016년에 제안된 Grassl et al.의 AES 양자 공격 연구[5]라는 점을 고려해야 한다. 왜냐하면 해당 연구는 최초의 AES 블록 암호 양자 공격 연구라는 의미를 가지나, AES의 양자 회로가 매우 비효율적으로 구현되었기에 공격 비용이 매우 높게 평가되어 있기 때문이다.

또한 NIST의 양자 후 보안요구사항 문서[4]에는, 추후에 공격 비용을 크게 감소시킨 양자 공격이 등장하는 경우, 본 문서에서 추정한 비용(2^{170}, 2^{233}, 2^{298})이 보수적으로 간주되어야 한다고 언급되어있다. 이후 비용을 감소시킨 다양한 AES 양자 회로 구현 연구들이 등장하였기 때문에[6], Table 4에서 SPARKLE에 대해 평가되는 양자 후 보안 강도들은 모두 보수적으로 평가하여야 한다. AES와 경량암호의 양자 회로 비용 추정 연구들[6, 10]에도 NIST의 추정 비용을 보수적 지표로 고려해야 한다고 제시되어 있다.

이에 따라 AES에 관한 최신 Grover 공격 비용 추정 연구[17]에서 제시한 비용(2^{157}, 2^{222}, 2^{286})을 기반으로 SPARKLE에 대한 양자 후 보안 강도를 평가해 본 결과, 네 가지 파라미터 모두 키 크기 기준 적정 보안 강도를 획득하였음을 확인하였다.

5. 결 론

가까운 미래에 등장할 것으로 예상되는 고성능의 양자 컴퓨터로부터 안전한 암호 시스템을 구축하기 위한 가장 확실한 방법은 사전에 다양한 암호 알고리즘에 대한 양자 후 안전성을 평가하는 것이다. 본 논문에서는 경량 암호 SPARKLE의 AEAD군인 SCHWAEMM 알고리즘을 양자 회로로 최적화 구현하였고, Grover 알고리즘 적용을 위해 필요로 하는 비용을 추정하여 NIST가 제시한 보안 요구사항을 기준으로 양자 후 보안성을 평가하였다. 해당 기준에 따르면 논문에서 제시한 양자 회로는 가장 낮은 보안 강도인 Level-1을 달성하지 못하였다. 그러나 NIST가 제시한 추정 비용을 보수적인 지표로 고려하고, 비용을 감소시킨 AES 회로의 추정비용을 기준으로 본다면 구현한 SPARKLE 양자 회로는 해당 보안 강도를 만족했다고 볼 수 있다.

향후 연구 방향으로는 첫째, SPAKRKLE 외의 NIST 경량암호 공모전의 최종 후보 알고리즘들을 양자 회로로 최적화 구현하고 이에 대한 Grover 공격 비용을 분석한 뒤 양자 후 보안 강도를 평가하는 것이다. 두 번째는 관련 연구에서 소개한 Draper의 Carry Look-ahead 양자 덧셈기 외에도 VBE 덧셈기 등 다양한 덧셈기들을 양자회로로 구현하고 적용하여 가장 최적화된 방법을 찾는 것이다. 이는 다가오는 양자 컴퓨팅 시대에 대비하는 안전한 IoT 시스템의 구축은 암호 학계에 있어 중요한 향후 계획이 될 것으로 사료된다.

References

[1] NIST, “Post-Quantum Cryptography Selected Algorithm 2022,” [internet], https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[2] NIST, “Post-Quantum Cryptography Round 4 Submissions,” [internet], https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[3] L. K. Grover, “A fast quantum mechanical algorithm for database search,” *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pp.212-219, 1996.

[4] NIST, “Submission requirements and evaluation criteria for the post-quantum cryptography standardization process,” [internet], https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[5] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, “Applying Grover’s algorithm to AES: Quantum resource estimates,” Post-Quantum Crypto Graphy, PQCrypto'16, LNCS, 9606, pp.29-43, 2016.

[6] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, “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.

[7] R. Anand, A. Maitra, and S. Mukhopadhyay, “Grover on SIMON,” *arXiv:2004.10686*, 2020.

[8] K. B. Jang, G. J. Song, H. J. Kim, H. D. Kwon, H, J. Kim, and H. J. Seo, “Efficient implementation of PRESENT and GIFT on quantum computers,” *Applied Sciences*, Vol.11, No.11, pp.4776, 2021.

[9] K. B. Jang, G. J. Song, H. D. Kwon, S. W. Uhm, H. J. Kim, W. K. Lee, and H. J. Seo, “Grover on PIPO,” *Electronics*, Vol.10, No.10, pp.1194, 2021.

[10] A. Baksi, K. B. Jang, G. J. Song, H. J. Seo, and Z. Xiang, “Quantum implementation and resource estimates for rectangle and knot,” *Quantum Information Processing*, Vol.21, No.7, 2021.

[11] C. Beierle et al., “Schwaemm and esch: Lightweight authenticated encryption and hashing using the Sparkle permutation family,” *NIST round*, 2, 2019.

[12] B. I. Kim, K. S. Min, and J. Heo, “Hamiltonian path problem approach using Grover search algorithm,” *The Journal of Communications and Networks*, Vol.2020, No.8, pp.52-53, 2020.

[13] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," *arXiv preprint quant-ph/0410184*, 2004.
[14] T. G. Draper, S. A. Kutin, E. M. Rains, and K. M. Svore, "A logarithmic-depth quantum carry- lookahead adder," *arXiv preprint quant-ph/0406142*, 2004.
[15] H. Thapliyal, H. V. Jayashree, A. N. Nagamani, and H. R. Arabnia, "Progress in reversible processor design: A novel methodology for reversible carry look-ahead adder," In: Transactions on Computational Science XVII. Springer, Berlin, Heidelberg, pp.73-97, 2013.
[16] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," *arXiv preprint arXiv:0910.2530*, 2009.
[17] K. B. Jang, A. Baksi, G. J. Song, H. J. Kim, H. J. Seo, and A. Chattopadhyay, "Quantum Analysis of AES," *Cryptology ePrint Archive*, 2022.

양 유 진

https://orcid.org/0000-0002-9007-2280
e-mail : yujin.yang34@gmail.com
2022년 한성대학교 IT융합공학부(학사)
2022년 ~ 현 재 한성대학교 IT융합공학과 석사과정
관심분야 : 정보보안, 양자컴퓨터

장 경 배

https://orcid.org/0000-0001-5963-7127
e-mail : starj1023@gmail.com
2019년 한성대학교 IT응용시스템공학부 (학사)
2021년 한성대학교 IT융합공학부(석사)
2021년 ~ 현 재 한성대학교 정보컴퓨터공학과 박사과정
관심분야 : 정보보호, 암호, 양자컴퓨터

김 현 지

https://orcid.org/0000-0001-9828-3894
e-mail : khj1594012@gmail.com
2020년 한성대학교 IT융합공학부(학사)
2022년 한성대학교 IT융합공학과(석사)
2022년 ~ 현 재 한성대학교 정보컴퓨터공학과 박사과정
관심분야 : 정보보호, 인공지능

송 경 주

https://orcid.org/0000-0002-4337-1843
e-mail : thdrudwn98@gmail.com
2021년 한성대학교 IT융합공학부(학사)
2021년 ~ 현 재 한성대학교 IT융합공학과 석사과정
관심분야 : 양자컴퓨터, 정보보안

임 세 진

https://orcid.org/0000-0001-7186-8104
e-mail : dlatpwls834@gmail.com
2022년 한성대학교 컴퓨터공학부(학사)
2022년 ~ 현 재 한성대학교 IT융합공학과 석사과정
관심분야 : 인공지능 보안, 정보보안

서 화 정

https://orcid.org/0000-0003-0069-9061
e-mail : hwajeong84@gmail.com
2010년 부산대학교 컴퓨터공학과(학사)
2012년 부산대학교 컴퓨터공학과(석사)
2016년 부산대학교 컴퓨터공학과(박사)
2016년 ~ 2017년 싱가포르 과학기술청 연구원
2017년 ~ 현 재 한성대학교 IT융합공학부 조교수
관심분야 : 정보보호, 암호구현