Grover's Algorithm을 통한 S-DES 공격 기법

이상헌*, 조민규*

*고려대학교 (학부생)

Cryptanalyzing S-DES Using Grover's Algorithm

Sangheon Lee*, Mingyu Cho*

*Korea University (Undergraduate)

요 약

본 논문에서는 양자 알고리즘인 Grover's algorithm을 사용하여 대칭키 암호시스템인 S-DES에 대한 Known Plaintext Attack을 수행한다. 이 과정에서 S-Box 계산을 수행하는 방법을 제시하며, 필요한 qubit의 수와 양자 논리 게이트의 수를 계산한다.

I. 서론

Grover's algorithm은 정렬되지 않은 N개의 데이터 중에서 원하는 성질을 만족하는 원소를 높은 확률로 $O(\sqrt{N})$에 찾아낼 수 있는 양자 알고리즘이다.[1] Grover's algorithm은 원하는 성질을 만족하는 원소에 대해 1을 반환하고, 아니면 0을 반환하는 가역적인 오라클(oracle) 함수만 있으면 적용이 가능하다. 행렬곱 검증문제나 최소신장트리 계산 등 다양한 문제에 Grover's algorithm을 도입함으로써 고전 컴퓨팅 체계보다 나은 시간복잡도를 이끌어낸 경우가 많다. [2]

때문에, AES-256 등의 기존의 암호화 알고리즘이 Grover's algorithm을 통한 양자 컴퓨팅에서 안전한지 판별하는 연구가 진행되고 있다. 하지만 현대 양자 컴퓨팅 기술로는 AES-256 등의 암호화 알고리즘을 물리적 구현은 커녕 시뮬레이션을 진행하는 것조차 불가능하다.

본 논문에서는 DES의 간소화된 형태인 S-DES[3]를 양자 컴퓨팅 측면에서 이론적으로 분석한다. 선행 연구[4]와 비교해서 구현체에 대한 양자 논리 게이트의 개수를 엄밀하게 계산하였으며, S-Box를 구성하는 다른 방법을 제안한다. 이어, Microsoft Q#으로 Grover's algorithm을 적용한 S-DES에 대한 Known Plaintext Attack을 구현하였다. 해당 구현체에서 사용한 qubit 개수에 따른 실행 시간 및 메모리를 분석하고 올바르게 작동함을 확인한다.

II. 배경 지식

2.1 Grover's Algorithm

Fig. 1은 Grover's algorithm의 과정을 나타낸다. Grover iteration이라 불리는 2번 과정을 반복하면서 원하는 특성을 지닌 기저의 위상의 부호가 반대가 되며, 대칭변환을 통해 차이가 증폭된다. 가능한 상태의 개수가 N개일 때,

$\frac{\pi}{4}\sqrt{N}$번 수행하면 $1-\frac{1}{N}$의 확률로 원하는 성질을 지닌 상태를 관찰할 수 있다. Grover's algorithm을 통해 얻을 수 있는 시간복잡도는 최적인 것이 알려져 있다. [5] 또, 성질을 만족하는 원소가 M개면 Shor's algorithm 등을 추가적으로 적용하며 $\frac{\pi}{4}\sqrt{\frac{N}{M}}$번 시행하면 $1-\frac{M}{N}$의 확률로 원하는 상태를 관찰할 수 있다. [6]

Algorithm 1. Grover's Algorithm

Input: A quantum oracle O such that $O(x_i)=1$ if $x_i \in A$ otherwise 0, and n qubits initialized to state $|0\rangle$.

Output: $x_i \in A$

1. Apply Hadamard operation to all qubits: $H^{\otimes n}|0\rangle^{\otimes n} = |\psi\rangle$
2. Apply Grover iteration $[(2|\psi\rangle\langle\psi| - I)O]^R \approx |x_i\rangle$ where $R \approx \frac{\pi}{4}\sqrt{2^n}$
3. Measure n qubits

Fig. 1. Grover's Algorithm.

2.2 S-DES

S-DES는 DES의 간소화된 버전으로, 키 사이즈와 라운드의 횟수 등이 손으로 계산할 수 있는 수준으로 줄어들었으나 기존 DES의 성질을 일부 보유하고 있는 암호화 알고리즘이다. [7] 키 생성과 암호화의 두 과정으로 나뉜다.

키 생성에서는 10-bit 키를 입력으로 받아 순환 시프트와 순열을 이용하여 두 개의 8-bit 라운드 키를 계산한다.

8-bit 평문은 총 두 번의 라운드를 통해 암호화되는데, 각 라운드별로 S-Box를 통과하여 암호화를 진행된다.

III. S-DES 양자 회로 분석

S-DES를 오라클 함수로 사용하기 위해 이를 가역 양자 회로로 구현해야한다. S-DES의 구조상 IP, P8 등 대부분이 순열을 통한 재배치로 구성되어 있기 때문에, 구현 과정에서는 재배열 등을 통해 qubit을 건들지 않는 경우가 많다. EP의 경우 qubit 4개를 8개인 것처럼 계산해야 하지만, 두 S-Box가 독립적이기 때문에 개별적으로 계산하고 역연산을 취하는 방법을 택할 수 있다. S-Box는 그 기능상 혼돈(confusion)의 특성을 이용해 키와 암호문의 관계를 줄이기 때문에[10] 간단한 논리식으로 표현되지 않아 많은 연산이 소요된다.

3.1 S-Box

3.1.1 Brute-forcing S-box

S-Box의 각 입력별로 아래 Fig. 2와 같은 회로를 사용하여 S-Box의 출력을 만들어낼 수 있다. Fig. 2는 입력으로 4가 주어질 때 출력이 3인 경우에 대한 양자 S-Box이다.

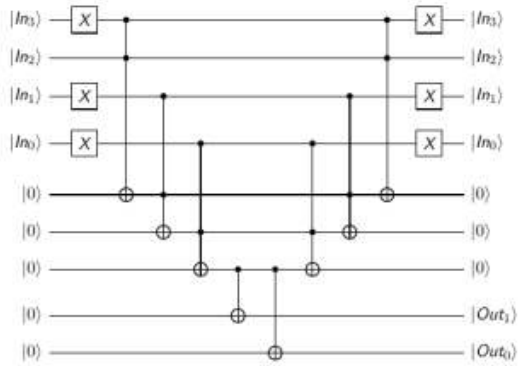

Fig. 2. Quantum S-DES S-Box circuit with 3 ancilla qubits.

[4]에서는 3번의 Tofolli gate와 3개의 ancilla qubit을 추가하는 식으로 S-Box를 구현하였다. 그러나 여러 개의 qubit에 대해 한 번에 controlled 연산을 진행할 수 있으면 추가 qubit 없이 Fig. 3 처럼 회로를 바꿀 수 있다.

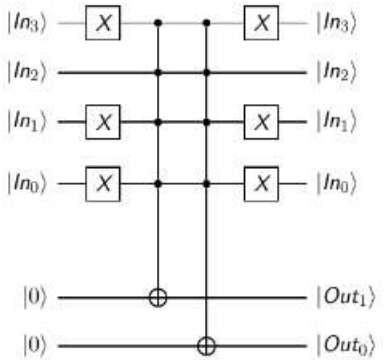

Fig. 3. Quantum S-DES S-Box without any ancilla qubits.

3.1.2 Quine-McCluskey Algorithm

Quine-McCluskey algorithm[8][9] 은 부울

대수에서 사용되는 논리 게이트의 수를 최소화하는 기법이다. 위 Fig. 3의 S-Box Lookup table에 Quine-McCluskey algorithm을 적용시켜 And, Or, Not을 사용한 논리식으로 나타낼 수 있다. 이렇게 구성한 회로를 위와 같은 방법으로 양자 회로를 구성하면 사용되는 게이트의 수를 줄일 수 있지만, 논리식에 대한 전처리가 필수적이고 추가 qubit을 할당해야 한다는 단점이 있다.

3.2 Qubit의 개수

초기에 구상했던 양자 S-DES 구현체의 qubit 사용처는 Table. 1과 같다.

qubit 사용처	qubit 개수
암호화 과정에서의 평문	10
암호화 과정에서의 키	8
EP를 통해 생성되는 qubit	8
S-Box 결과 저장용 qubit	4
S-Box에서의 ancilla qubit	3
암호화 오라클 결과 qubit	1

Table 1. Initial usage of qubits of quantum S-DES implementation.

그러나 시뮬레이션의 특성상, qubit이 하나 추가될 때마다 수행시간과 메모리가 2배 정도 증가하였기 때문에 qubit의 개수를 최소화할 필요가 있었다. 3가지 방법으로 qubit의 개수를 최소화할 수 있었다.

첫째로 EP를 통해 만든 8개의 qubit을 평문을 직접 이용한 방식으로 제거하였다. 둘째로 두 S-Box가 독립적이라는 사실을 이용해 저장용 qubit을 2개로 줄였다. 마지막으로 Microsoft Q#의 문법을 이용하여 S-Box에서의 ancilla qubit을 제거할 수 있었다.

최종적으로 S-DES 양자 회로에서 사용되는 qubit의 개수는 기존의 34개에서 13개를 줄인 21개로, 목록은 Table 2와 같다.

qubit 사용처	qubit 개수
암호화 과정에서의 평문	10
암호화 과정에서의 키	8
S-Box 결과 저장용 qubit	2
암호화 오라클 결과 qubit	1

Table 2. Optimized usage of qubits of quantum S-DES implementation.

IV. 실험 결과

Microsoft가 만든 양자 컴퓨팅 프로그래밍 언어인 Q#을 통해 S-DES 암호화 함수를 만들고, Grover's algorithm을 이용하여 구현하였으며, 평문과 그에 대응되는 암호문을 입력으로 하여, 암호화 키를 구하는 Known-Plaintext Attack을 시도하였다.

4.1 Qubit의 개수에 따른 실행 시간과 메모리

Qubit의 개수는 Microsoft Q#의 구동 시간을 결정하는 중요한 요소이다. 다양한 구현체에서 S-DES의 수행 속도를 측정해보았으며 그 결과는 Table 3과 같다.

qubit 개수	수행 시간	메모리
32개 (비가역)	비정상 종료	48GB 할당 후 비정상 종료
26개 (비가역)	2분 25초	1.8 GB
24개	3분 6초	272.4 MB
21개	8.6초	49.3 MB

Table 3. Running time and memory usage of each quantum S-DES implementation.

처음 두 구현체는 S-Box와 암호화 최종 단계에서 qubit을 관측하였기 때문에 빠르지만 비가역적이다. 32개의 qubit을 할당했을 때는 프로그램이 비정상 종료되었으나 6개를 줄이자 프로그램이 정상 종료되었다. 관측을 하면 역과정을 거칠 필요가 없기에 빠르지만 Grover's algorithm을 사용할 수가 없어 S-DES 구현 검증에만 사용하였다.

세 번째 구현체는 S-Box에서 ancilla qubit 3개를 이용하였으며 네 번째는 Microsoft Q#의 controlled functor를 이용해 qubit을 줄였다. qubit 개수는 3개밖에 차이가 나지 않지만 S-Box를 통해 결과 qubit을 만들고 다시 역과정을 거쳐 기존 평문 qubit을 초기화해줘야 하기 때문에 수행 시간이 8배보다 크게 차이가 난 것으로 보인다.

4.2 양자 논리 회로의 복잡도

Qubit의 개수만큼 중요한 요소 중 하나가 양자 논리 회로(quantum gate)의 개수이다.

S-Box를 제외한다면, 데이터를 복사하고 사

용하는데에 CNOT gate 40개가 필요하다. S-Box의 경우 각각이 라운드 수만큼 필요하다. 따라서, Table 4와 같은 수의 Pauli-X와 Toffoli gate가 필요하다.

Pauli-X의 경우 수가 유사하지만, Toffoli gate의 수가 반 이상 줄어들었다는 점을 고려한다면 유의미한 속도 증가를 이끌어낼 수 있을 것으로 예상된다. 특히 DES의 경우 6개의 S-Box가 16라운드 동안 필요하기에, 더욱 큰 속도 증가를 이끌어낼 수 있으리라 예상된다.

Method	Pauli-X	Toffoli
Brute-force	188	66
Quine-McCluskey	172	26

Table 4. Required number of Pauli-X and Toffoli(CCNOT) gates for each S-Box method.

V. 결론 및 제언

S-DES에 대한 Known-Plaintext Attack을 위해, 양자 S-DES 오라클을 만들고 Grover's algorithm을 사용하여 암호화 키를 찾아보았다. 본 연구에서는 S-DES를 활용하였으나, 이와 같은 접근이 DES 등 다른 대칭키 암호 오라클의 구축에 활용할 수 있을 것으로 생각된다.

S-DES는 임의의 평문과 암호문 쌍이 주어졌을 때 대응되는 키의 개수가 최소 1개에서 최소 17개로 균일하지 않으며, 일반적인 암호화 알고리즘도 이럴 것으로 예상된다. 다만 Grover iteration의 횟수를 조정하여 오라클의 해의 개수 M이 알려져 있지 않아도 $O(\sqrt{\frac{N}{M}})$번 오라클을 호출하여 해를 구할 수 있는 방법이 존재하기에[6] 어렵지 않게 확장할 수 있을 것으로 보인다.

[참고문헌]

[1] Grover, Lov K. "A fast quantum mechanical algorithm for database search." Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996.

[2] Coles, Patrick J., et al. "Quantum algorithm implementations for beginners." arXiv preprint arXiv:1804.03719 (2018).

[3] Schaefer, Edward F. "A simplified data encryption standard algorithm." Cryptologia 20.1 (1996): 77-84.

[4] Almazrooie, Mishal, et al. "Quantum exhaustive key search with simplified-DES as a case study." SpringerPlus 5.1 (2016): 1494.

[5] Bennett, Charles H., et al. "Strengths and weaknesses of quantum computing." SIAM journal on Computing 26.5 (1997): 1510-1523.

[6] Boyer, Michel, et al. "Tight bounds on quantum searching." Fortschritte der Physik: Progress of Physics 46.4 5 (1998): 493-505.

[7] Stallings, William. Cryptography and network security, 4/E. Pearson Education India, 2006.

[8] Quine, Willard V. "The problem of simplifying truth functions." The American mathematical monthly 59.8 (1952): 521-531.

[9] Quine, Willard V. "A way to simplify truth functions." The American Mathematical Monthly 62.9 (1955): 627-631.

[10] Shannon, Claude E. "Communication theory of secrecy systems." Bell system technical journal 28.4 (1949): 656-715.