Quantum Speedup on Exhaustive-search Attacks on Cryptosystems

Week 7 Report for Class 75

2017320009 Sangheon Lee
2017320023 Mingyu Cho

May 25, 2020

Table of Contents

Backgrounds

Mathematical Backgrounds

- Complex numbers
  - Complex plane $a + bi = (a, b)$
  - Complex polar $\rho e^{\phi i} = \rho \cos \phi + \rho \sin \phi i$
- 2-dimensional Hilbert space
  - A complex vector space, utilizing conjugate transposes.
  - When constricted to $\mathbb{R}$, same as $\mathbb{R}^2$ vector space.
- Most of the time on Week 4 was spent on understanding the mathematical backgrounds on 2-dimensional Hilbert spaces and special matrices which can only be considered on $\mathbb{C}$.
- Additionally, analyzing Grover's Algorithm implementation in Microsoft Q$\#$ was done.

Qubits

Qubits

- Quantum Bit(Qubit for short) is a probabilistic vector of information.
- $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- A general qubit is represented as $u = \alpha |0\rangle + \beta |1\rangle$

Three Main Types of Operation

- Creation
- Reversible Operation
- Measurement

Creation

- As simple as creating $|0\rangle$ or $|1\rangle$.

Reversible Operation

- Represented using unitary matrix
  - $U^* U = UU^* = I$, where U^* is a conjugate transpose of U
- Two frequently used operations
  - X-gate $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ (so-called NOT gate)
  - Hadamard Gate $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
  - Hadamard Gate creates a Quantum Superposition

Measurement

- A non-deterministic(probabilistic) measure of a qubit u.
- For a vector $u = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ where $\|u\| = 1$
  - returns "true" with probability $\|\alpha\|^2$, and u becomes $|0\rangle$
  - returns "false" with probability $\|\beta\|^2$, and u becomes $|1\rangle$
- Destroys quantum superposition; often called "destructive".

Grover's Algorithm

Overview

- Grover's algorithm can find a specific state satisfying some condition among $N = 2^n$ candidates in $O(\sqrt{N})$ time, compared to classical runtime complexity $O(N)$.
- Grover's algorithm exploits qualities of quantum amplitudes to gain advantage of probability seperation.
- It can brute-force 128-bit symmetric cryptographic key in roughly 2^{64} iterations.

Algorithm

Input:

- A quantum oracle $\mathcal{O}$ which performs the operation $\mathcal{O}|x\rangle = (-1)^{f(x)}|x\rangle$, where $f(x) = 0$ for all $0 \leq x < 2^n$ except x_0, for which $f(x_0) = 1$.
  - Such quantum oracle is viable, and takes $O(1)$ time.
- n qubits initialized to the state $|0\rangle$

Output: x_0, in runtime $O(\sqrt{2^n})$ with error rate $O(\frac{1}{2^n})$

Algorithm

Procedure:

1. $|0\rangle^{\otimes n}$ (initial state)
2. $H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle = |\psi\rangle$ (Hadamard transform)
3. $[(2|\psi\rangle\langle\psi| - I)\mathcal{O}]^R|\psi\rangle \approx |x_0\rangle$ (Grover iteration for $R \approx \frac{\pi}{4}\sqrt{2^n}$ times)
4. x_0 (measure)

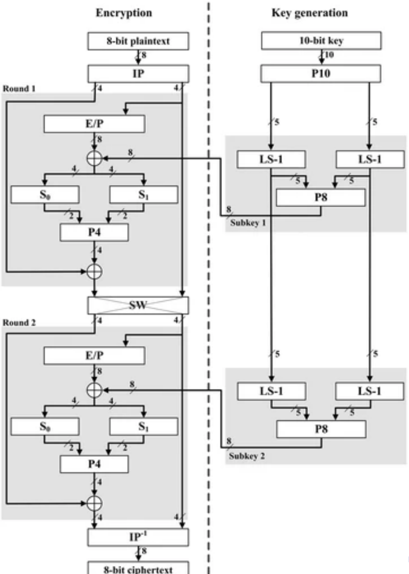
Grover iteration in a nutshell: negate the amplitude of the desired state, followed by 'diffusion transform' which increases the amplitude of the desired state and lower the others.

Quantum S-DES Oracle

S-DES

- Simplified DES with 2 rounds
- Structure similar to DES but simplified with 10-bit key and 8-bit plaintext.
- Quantum oracle needs to be reversible, of which S-DES is (normally) not.

Structure of S-DES



Quantumizing S-DES

- Most parts are permutations or compressions; no problems here.
- Two parts poses a challenge:

1. S-Boxes
   - Consists of lookup table: not ideal for our situation.
   - ~~Broken the S-Boxes into fundamental and/or/not-gates~~
   - Actually, the brute-force method(of checking each cases) enables reusage of ancilla bits, thereby reducing the number of required qubits!
   - Classic gates are not reversible; use CNOT/CCNOT gates to make them reversible.
2. Expansion
   - Due to No-cloning theorem, directly copying a qubit is not possible.
   - Use XOR operation to copy the information.

Quantum S-DES Implementation

S-DES Obstacles

- Expansion : Directly clone with XOR twice
  - With extra 8 qubits, we can explicitly use expanded plaintext
- S-Box : Passing all the candidates
  - Denote S-Box function as $f\colon \{0,1\}^4 \to \{0,1\}^2$
  - If the dealing plaintext has a basis for $x \in \{0,1\}^4$, apply XOR(NOT) to result S-Box qubit

Quantum S-DES Implementation

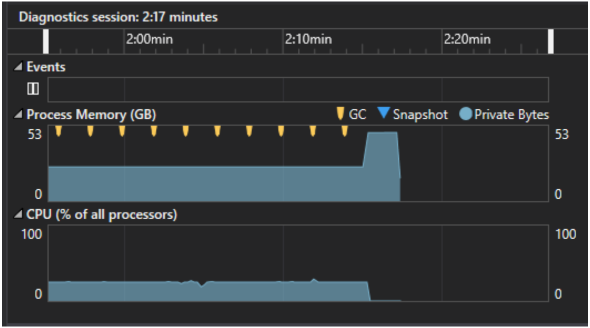
Initially, 34 explicit qubits were used in total.

- 10 qubits for input key
- 1 qubit for making an encryption oracle
- 8 qubits for (intermediate) plaintext
- 8 qubits for expanded plaintext
- 4 qubits for storing S-Box result
- 3 qubit for S-Box ancilla

S-DES implementation is written in Microsoft Q#. Will it run?

Quantum S-DES Implementation

Yes, and no.



After allocating 48GB of RAM,
`System.Runtime.InteropServices.SEHException` was raised, indicating an out-of-memory error.

Quantum S-DES Implementation

Reducing the number of qubits was necessary...!

- 10 qubits for input key
- 1 qubit for making an encryption oracle
- 8 qubits for (intermediate) plaintext
- 8 qubits for expanded plaintext
- 4 qubits for storing S-Box result
- 3 qubit for S-Box ancilla

Quantum S-DES Implementation

Reducing the number of qubits was necessary...!

- 10 qubits for input key
- 1 qubit for making an encryption oracle
- 8 qubits for (intermediate) plaintext
- ~~8 qubits for expanded plaintext~~
- 4 qubits for storing S-Box result
- 3 qubit for S-Box ancilla

We can directly use plaintext qubits to create result qubits from S-Box. After then we can undo all the operations (permutation, *etc.*) which altered plaintext.

Quantum S-DES Implementation

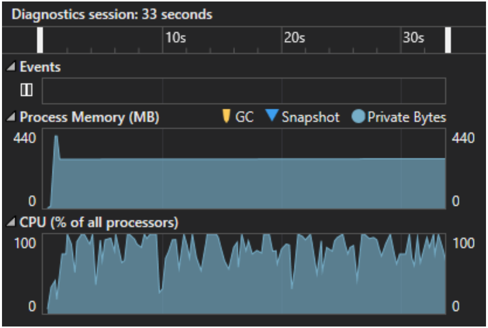
Reducing the number of qubits was necessary...!

- 10 qubits for input key
- 1 qubit for making an encryption oracle
- 8 qubits for (intermediate) plaintext
- ~~8 qubits for expanded plaintext~~
- ~~4~~ 2 qubits for storing S-Box result
- 3 qubit for S-Box ancilla

We can directly use plaintext qubits to create result qubits from S-Box. After then we can undo all the operations (permutation, *etc.*) which altered plaintext.
Additionally, each S-Box operation is independent. Thus we can reuse S-box result qubits after re-initialization.

Quantum S-DES Implementation

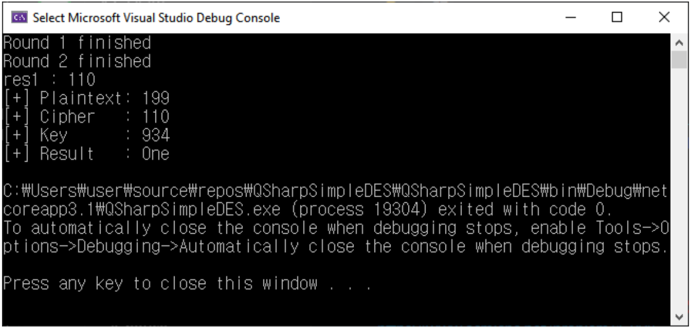
After a bit of debugging and comparing with Python Implementation...



It ran well within 40s (before the second optimization, it took 4x time and memory)

Quantum S-DES Implementation

After a bit of debugging and comparing with Python Implementation...



Expected cipher matched with Python S-DES implementation output.

Quantum S-DES Implementation

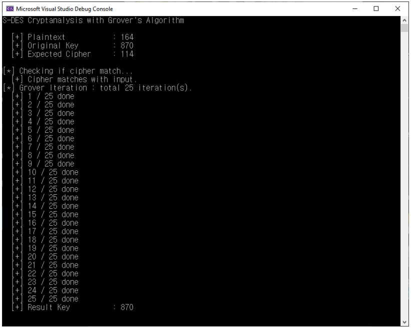
However, applying Grover's was not simple as it looked

- The oracle should be adjoint, but current version includes measurement (in result ciphertext and S-Box)
- How to create an oracle qubit rather than measuring the result ciphertext?
    - perform CNOT gate 8 times (with 7 more qubits) : infeasible
    - `Controlled` functor : doable
    - All-one oracle : similar (which is faster?)

Quantum S-DES Implementation

- Measurement was wiped out smoothly with the cost of execution speed
- Each round need to be reversed, and so does the S-Box, increased runtime about 4x
- Qubit optimization in S-Box offset this demerit
  - Used the same 'all-one oracle' trick

Quantum S-DES Implementation



Expected key matched with original key (with runtime 19 min 31 sec, performing each S-DES encryption within 45s on average).

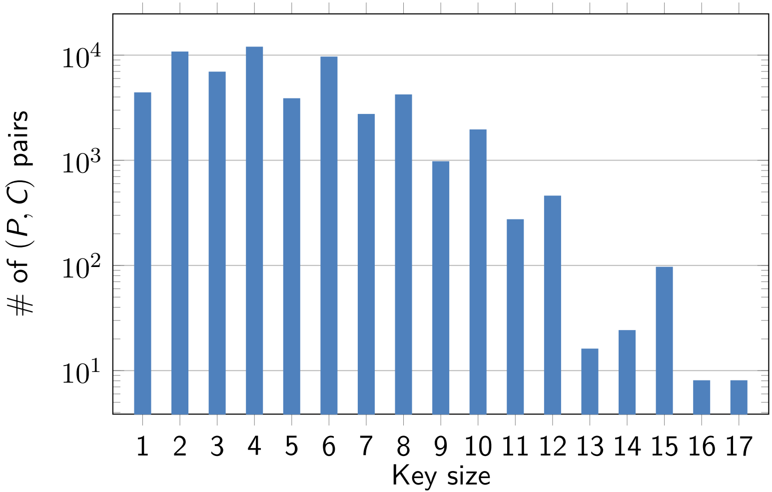
Quantum S-DES Testing

Prerequisites

- Let $SDES(P, K) = C$ be a function s.t. encrypting plaintext P with S-DES and key K yields C.
- Let $f\colon \{0,1\}^8 \times \{0,1\}^8 \to \mathcal{P}(\{0,1\}^{10})$ be a function such that $f(P, C) = \{K \in \{0,1\}^{10} \mid SDES(P, K) = C\}$.
- Since key size is 4x of cipher size, we guessed that for arbitrary P and C, $|f(P, C)|$ is likely to be 4.

Prerequisites : Key Size

...and we were wrong!



Quantum S-DES Testing : Testing with #Key = 1

If $(P, C) = (10100100, 00110011)$. Then $f(P, C) = \{1101100110\}$.

- Program yielded correct key.
- By dumping key qubits, we can observe the probability of each basis.
  - At first, the probability of answer basis is 0.008766 while the others are 0.000969.
  - The gap widens next turn: 0.024224 and 0.000954.
  - Ultimately it becomes 0.999461 and 0.000001 (at 25th iteration).

Quantum S-DES Testing : Testing with #Key $= 4$

If $(P, C) = (11000111, 00010100)$. Then $|f(P, C)| = 4$.

- Program yielded wrong key.
- By dumping key qubits, we can observe the probability of each basis.
  - At first, the probability of answer basis is 0.008698 while the others are 0.000946.
  - The gap widens next turn: 0.023659 and 0.000888.
  - It peaks at 12th iteration: 0.249987 and 0.000000.
  - However it goes back next turn: 0.246547 and 0.000014
  - Ultimately the probability of answer basis becomes lower than the others: 0.000575 and 0.000978.
- Although the algorithm and the implementation is correct, what happened here?

Current Works and Future Plans

Work in Progress(Mingyu Cho)

- Implemented a non-quantum S-DES(in python) for testing the quantum version.
- Optionally implement Brute-force attack on S-DES?
- Started preliminary analysis on the gate applications
- Looking for possible speedups on the quantum side to aid the simulation

Work in Progress(Sangheon Lee)

- Possibly reduce some quantum gates
- Analyze complexity of each gate
- Search for internal dumping library

Future Plans

- Construct and test S-DES and apply Grover's.
  - Since S-DES consists of various components, all of these must be implemented carefully and in order
- Hope to import Microsoft Q# implementation to qiskit (IBM Quantum Experience)
- Reduce complexity of quantum gates if possible (as Prof. Hong suggested)