APPLYING QUANTUM SEARCH TO A KNOWN-PLAINTEXT ATTACK ON TWO-KEY TRIPLE ENCRYPTION

Phaneendra H.D. , Vidya Raj C. , Dr. M.S. Shivakumar
Assistant Professor, Department of Computer Science and Engineering, The National Institute of Engineering, Mysore, Karnataka, India

Assistant Professor, Department of Computer Science and Engineering, The National Institute of Engineering, Mysore, Karnataka, India

Principal, The National Institute of Engineering, Mysore, Karnataka, India

Abstract: The process of disguising a plaintext into ciphertext is called encryption and back into plaintext is called decryption. A cryptographic algorithm, is also called a cipher, is the mathematical function used for encryption and decryption. Many algorithms are available for this purpose. Triple DES is such an algorithm. Encryption using triple DES is possible in two different ways; they are triple DES with two keys and triple DES with three different keys. Cryptanalysis can be used to recover the plaintext of a message from the ciphertext without access to or knowing the key. Exhaustive key search remains the most practical and efficient attack on Triple DES with two keys. The principles of quantum mechanics can be used to build and analyze a quantum computer and its algorithms. Quantum searching is one such algorithm. The key search in Triple DES with two keys is possible using quantum search algorithm, which is more efficient compare to any other methods. In this paper we are presenting how quantum search can be used to crack Triple DES with two keys searching for a key.

Keywords: Quantum mechanics, Quantum algorithm, qubits, Quantum search, 3DES, exhaustive search

1. INTRODUCTION

1.1 Quantum computation

Quantum Computation is the field of study, which focused on developing computer technology based on the principles of quantum theory. The aim of this paragraph is to make computer scientists to go through the barriers that separate quantum computing from conventional computing. We have introduced the basic principles of quantum computing and tried to implement it in applications like key searching for cracking cryptographic algorithms like DES, Double DES and 3DES. It is important for the computer science community to understand these new developments since they may radically change the way we think about computation, programming, and complexity [1]. The basic variable used in quantum computing is a qubit, represented as a vector in a two dimensional complex Hilbert space where $|0\rangle$ and $|1\rangle$ form a basis in the space. The difference between qubits and bits is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$ whereas a bit has only one state, either 0 or 1. It is also possible to form linear combination of states, often called superposition. The state of a qubit can be described by

$$|\psi\rangle = \alpha|0\rangle + ß|1\rangle \quad (1)$$

The numbers α and ß are complex numbers. The special states $|0\rangle$ and $|1\rangle$ are known as computational basis states. We can examine a bit to determine whether it is in the state 0 or 1 but we cannot directly examine a qubit to determine its quantum state, that is values of α and ß. When we measure a qubit we get either the result 0, with probability $|\alpha|^2$ or the result 1, with probability $|ß|^2$, where $|\alpha|^2 + |ß|^2 = 1$, since the probabilities must sum to one. Consider the case of two qubits. In two classical bits there would be four possible states, 00, 01, 10 and 11. Correspondingly, a two qubit system has four computational basis states denoted $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. A pair of qubits can also exist in a superposition of these four states, so the quantum state of two qubits involves associating a complex coefficient, sometimes called amplitude, with each computational basis state, which is given as

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \quad (2)$$

The logic that can be implemented with qubits is quite distinct from Boolean logic, and this is what has made quantum computing exciting by opening new possibilities [8].

1.2 Quantum algorithms

Quantum algorithms are based on the principles of quantum mechanics. They are different from classical computing in two specific features: superposition and entanglement. Superposition can transfer the complexity of the problem from a large number of sequential steps to a large number of coherently superposed quantum states. Entanglement is used to create complicated correlation's that permit the desired interference.

A typical quantum algorithm starts with a highly superposed state, builds up entanglement, and then eliminates the undesired components providing compact results. In classical systems, the time taken to do certain computations can be decreased by using parallel processors. To achieve an exponential decrease in time, it requires an exponential increase in the number of processors, and hence an exponential increase in the amount of physical space. However, in quantum systems the amount of parallelism increases exponentially with the size of the system. Thus, an exponential increase in parallelism requires only a linear increase in the amount of physical space. This property is called quantum parallelism [8][2][5].

Suppose we are given a map containing many cities, and wish to determine the shortest route passing through all the cities on the map. A simple algorithm to find this route is to search all possible routes through the cities, keeping a running record of which route has the shortest length. On a classical computer, if there are N possible routes, it takes O(N) operations to determine the shortest route using this method. But quantum search algorithm enables this search method to be sped up substantially, requiring only $O(\sqrt{(N)})$ operations.

The quantum search algorithm is general in the sense that it can be applied far beyond the route finding example just described to speed up many (though not all) classical algorithms that use search heuristics. Thus given a search space of size N, and no prior knowledge about the structure of information in it, if we want to find an element in search space satisfying a known property, then this problem requires approximately N operations, but the quantum search algorithm allows it to be solved using approximately $\sqrt{(N)}$ operations[8][5].

2. TRIPLE DES

The Data Encryption Standard (DES) is a widely-used algorithm for encrypting data. It was developed by IBM under the name Lucifer. DES is a product block encryption algorithm (a cipher) in which 16 iterations, or rounds, of the substitution and transposition (permutation) process are cascaded. The block size is 64 bits, so that a 64-bit block of data (plaintext) can be encrypted into a 64-bit ciphertext. The key, which controls the transformation, also consists of 64 bits. Only 56 bits of these, however, are at the user's disposal; the remaining eight bits are used for checking parity. The actual key length used for encryption is therefore 56 bits. The same key is used for decryption [12]. DES is vulnerable to brute force attack, where key space can be searched for possible key to decrypt the message. Alternative is double DES. This scheme apparently involves a key length of 56 x 2 = 112 bits, resulting in a dramatic increase in cryptographic strength. Brute-force requires an exhaustive search of 2^{112} keys [12]. But this can be attacked by means of an algorithm known as meet-in-the middle attack.

The key which is used for encryption of the plaintext in DES and Double DES can also be found by means of using quantum search algorithm, sometimes called Grover's search algorithm. The algorithm enables search method to be sped up substantially requiring only O ($\sqrt{}$(N)) operations.

An obvious counter to these attacks is to use three stages of encryption with three different keys. This raises the cost of brute-force attack to 2^{168} keys, because it requires a key length of 56 x 3 = 168 bits. This also raises the known plain-text attack to 2^{112} keys.

Encryption using triple DES is possible in two different ways; they are triple DES with two keys and triple DES with three different keys.

2.1 Triple DES with two Keys

This method was proposed by Tuchman that uses only two keys. The method follows an encrypt-decrypt-encrypt (EDE) sequence as shown in the following figure 1. [3].

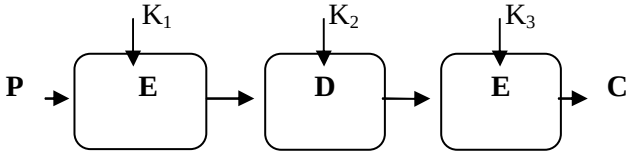


Figure 1. Triple DES with two Keys

The method operates on a block three times with two keys, with the first key, then with the second key, and finally with the first key again. That

means the sender first encrypt with the first key, then decrypt with the second key, and finally encrypt with the first key. The receiver decrypts with the first key, then encrypts with the second key, and finally decrypts with the first key.

$$C = E_{K1}(D_{K2}(E_{K1}(P)))$$
$$P = D_{K1}(E_{K2}(D_{K1}(C)))$$

The triple encryption with two keys is susceptible to chosen-plaintext attack and known-plaintext attack.

The chosen-plaintext attack requires an enormous amount of chosen-plaintext to mount. It requires 2^n time and memory (where n is the length of the key), and 2^m chosen- plaintexts. It is not very practical, but it becomes a weakness.

The known-plaintext attack, requires p known plaintexts, and assumes encryption is made using EDE (encrypt-decrypt-encrypt) mode. The algorithm for this is given by P.C. Van Oorschot and M.J. Wiener [3] [9].

3. THE PROPOSED METHOD USING QUANTUM SEARCH

The keys that are used for encryption of the plaintext in triple encryption with two keys can be found by means of using quantum search algorithm, sometimes called Grover's search algorithm. The algorithm enables search method to be sped up substantially requiring only O ($\sqrt{(N)}$) operations. The algorithm for this is as follows.

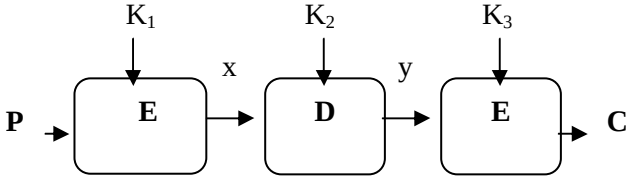


Figure 2. EDE mode.

1. We need to guess the first intermediate value, x (figure 2)
2. Then store, for each of the 256 possible K1, the second intermediate value, y, when the first intermediate value is x, using known plaintext:

$$y = D_{K1}(C)$$

Where C is the resulting ciphertext from a known plaintext.

(Resulting values need not be sorted, because quantum search works onto nonordered values)

Table 1. Table of intermediate values and key

y	K_1
.	.
.	.
.	.
.	.

3. Quantum search the table 1, for each of the 256 possible K2, elements with a matching second intermediate value, y:

$y = E_{K2}(x)$

If there is a match, then the corresponding key K_1 from table and the current value of K_2 are candidate values for the unknown keys (K_1, K_2).

(Search is possible with O ($\sqrt{(2^{K1})}$) for each K_2)

4. The probability of success is p/m, where p is the number of known plaintexts and m is the block size. If there is no match, try another x and start again from step 1.

This algorithm reduces the number of searches required for the keys.

The keys can be found with only $\sqrt{(2^{K1})}$ searches with the best case, and $2^{K2} * \sqrt{(2^{K1})}$ with the worst case. On an average $2^{K2}/2 * \sqrt{(2^{K1})}$ searches are required for finding the keys K_1 and K_2.

4. THE TIME AND SPACE ANALYSIS

In this section, we briefly summarize the running time and amount of memory required considering the algorithm given by P.C. Van Oorschot and M.J. Wiener [9] and our proposed algorithm.

There are three different cases arises for time and space analysis.

Case 1: The resulting values produced in step 2 will be stored onto a table, and the table need not be sorted. If this table is searched in step 3, the time required for a match is the order of 2^{56} at the worst case.

Case 2: The resulting values produced in step 2 will be stored onto a table, and the table is sorted. If this table is searched in step 3, the time required for a match is the order of O ($\log_2 2^{56}$). Here we have to consider the time required for sorting the table.

Case 3: The resulting values produced in step 2 will be stored onto a table, and the table need not be sorted. If this table is searched in step 3, the time required for a match is the order of O ($\sqrt{(2^{56})}$) (quantum search works onto nonordered values).

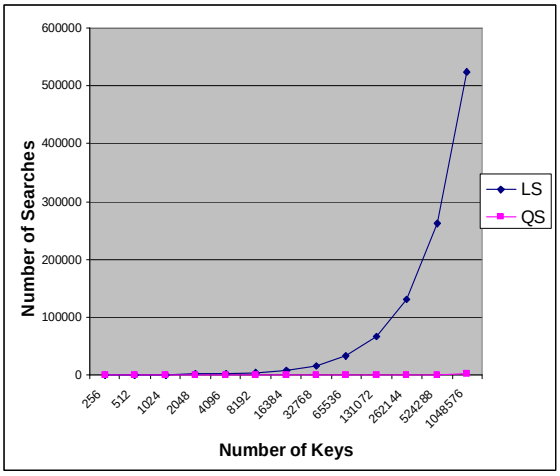
The space required to store resulting values in case 1 and case 2 are of the order of 2^{56} i.e., O(2^{56}).

The space required to store the resulting values in case 3 is difficult to specify, because quantum search works on quantum computers and it uses concept of superposition and quantum parallelism.

5. PERFORMANCE ANALYSIS BY SIMULATION.

Table 2. Comparison between Linear and Quantum search

No of Keys	Linear search (Non ordered values)	Quantum search (Non ordered values)
256	128	16
512	256	22
1024	512	32
2048	1024	45
4096	2048	64
8192	4096	90
16384	8192	128
32768	16384	181
65536	32768	256
131072	65536	362
262144	131072	512
524288	262144	724
1048576	524288	1024



Graph1. Comparison between Linear and Quantum search

Here the table 2 gives the comparison between use linear search and quantum search for key searching. We are not comparing binary search since it needs values are to be sorted before searching, and sorting takes considerable amount of time. The graph shows how quantum search out performs the linear search.

6. CONCLUSION

In this paper, we studied and analyzed quantum search algorithm based on quantum mechanics, by applying to a known-plaintext attack on two-key triple encryption. This algorithm works on unsorted list (step 2 of the algorithm), and provides a quadratic speed-up and the desired item is located with O ($\sqrt{(2^{K1})}$) queries (step 3 of the algorithm) with the best case. Theoretically it can be concluded that quantum search algorithm provide fast results by taking the help of quantum mechanics concepts like quantum parallelism and superposition. Quantum computing is a field in its infancy. When quantum computing was first systematically investigated, the main fear was that the natural world would not be able to realize any accuracy. These early concerns are now being overshadowed by the greater accomplishments in quantum computing. The consensus is that triple encryption with two keys, when used properly, is still secure. But a known-plaintext attack becomes easy looking at the research in quantum computation.

REFERENCES

1. Amardeep Singh and Sarbjeet Singh,(2003), Applying quantum search to automated test pattern generation for VLSI circuits, International Journal of Quantum Information,Vol.1 No. 1, 79-91.
2. Apoorva Patel, Quantum Database Search can do without Sorting, quant-ph/0012149.
3. Bruce Schneier,(2002), Applied Cryptography, Wiley International Pvt Ltd, New Delhi.
4. Grover L.K, (1996), A Fast Quantum mechanical Algorithm for Database Search, In proceedings of the 28th Annual ACM Symposium on the Theory of Computing,pp.212-219,quant-ph/9605043.
5. Grover L.K. (2002), Tradeoffs in the Quantum Search Algorithm, quant-ph/0201152.
6. Nielsen M and Chaung, I (2000), Quantum Computation and Quantum Information, Cambridge University press, Cambridge, United Kingdom.
7. P.C. Van Oorschot and M.J. Wiener A Known-Plaintext Attack on Two-Key Triple Encryption.Advances in Cryptology – EUROCRYPT 90 Proceedings.
8. Terry Rudolph and Dr. Lov Grover,(2002), Quantum Searching a classical database, quant-ph/0206066.
9. William Stallings, (2003), Cryptography and Network Security, Principles and Practices. Pearson Education, New Delhi.