International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 14 (2022) pp. 128-137
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Received: 14 October, 2021; Accepted: 6 February, 2022; Publish: 24 April, 2022

Quantum Cryptanalysis on A5/1 Stream cipher

Swamy Naidu Allu[1,2], Appala Naidu Tentu[2]

[1]Department of Computer Science & Engineering
Acharya Nagarjuna University, Guntur - 522510, (Andhra Pradesh) India.
alluswamynaidu33@gmail.com

[2] CR Rao Advanced Institute of Mathematics, Statistics and Computer Science,
University of Hyderabad Campus, Hyderabad - 500046, (Telangana) India.
naidunit@gmail.com

Abstract: **In this Paper, we present Quantum Cryptanalysis on A5/1. We focus on the fundamental query, *Are Symmetric Ciphers really insecure against quantum adversary*? We tried to understand this question by considering A5/1 symmetric key stream cipher. The hardware implementation of symmetric key Block cipher designs under combinational circuits, where symmetric key Stream cipher, in particular, LFSR based ciphers(A5/1 etc.) circuits belong to the sequential circuit group. In this paper we presented to get a clearer view about the quantum attack on A5/1 and hence on sequential circuit. By exploiting Grover's algorithm one can bypass the huge off-line computation required for classical Time/Memory/Data Trade-off attack. We have been focusing on practical implementation of this quantum attack in IBM quantum computer interface. We also implemented reduced version of Quantum A5/1(10-bit) using Qiskit programming and also estimated number of gates and working qubits required for full scale implementation of A5/1 cipher.**

Keywords: A5/1, IBMQ, Qubit, Symmetric key, LFSR, Grovers algorithm.

I. Introduction

In 1930 the fundemental model of classical computers was originally created by Alan Turing, Von Neumann and few other researchers. Anyway the model of PCs, that Turing or Neumann considered, are restricted by old style physical science and subsequently named as traditional PCs. Till the starting of twentieth century, biggest researchers authorized that Newtonian laws overseeing the movement of material bodies and Maxwell's hypothesis of electromagnetism are the principal spaces of physical science. In 1925 the disclosure of X-beams and electrons towards the finish of that century at last assisted the physicists with understanding quantum mechanics and after that they understood the classical mechanics.

In 1982 Richard Feynman[1] given the fundamental idea about a quantum computer or quantum simulator. Generally speaking that any quantum system contains greater than one particles can be understood by a Hilbert space. The dimension of this Hilbert space is exponentially large in the quantity of particles. In such a way that, one normally think that quantum system can effectively solve the problems that might require exponential time on a classical systems. In 1980's the introductory work done by Deutsch-Jozsa [2] and Grover [3] could analyze quantum ones that are exponentially quicker than the classical algorithms. In 1994 Peter Shor [4] invented in quantum model that discrete log and factorization problems can be accurately solved. This invertion is big impact in classical cryptography. More public key cryptosystems that are based on these hard problmes only. Online banking and whole internet communication also depends on the security of these. Thus, in public key cryptography, this allowed for basic building blocks in cryptography that can prevent the attacks even survival of quantum comuters. Economic quantum computers are still tricky. In 2012 Wineland and Haroche received Nobel prize for Physics for "ground-breaking experimental methods that enable measuring and manipulation of individual quantum systems". In 2016 Thouless, Haldane and Kosterlitz for "theoretical discoveries of topological phase transitions and topological phases of matter". These outcomes may have significance towards real applications of a quantum computer. Consequently it shows that this space of research is for sure one of the top preference in worldwide scientific community. In 2015, US security agency NSA published a report [5] where they clearly declared the urgency of quantum safe cryptographic protocols. This leads the researches to search some alternative solutions. In classical domain, there are Code based and Lattice based cryptosystems which are believed to be secure in quantum paradigm. Another avenue is Quantum Cryptography that warrants the security against quantum adversary.

In modern conditions, in quantum cryptography, two most prominent sub-domains are Quantum Key Distribution (QKD) [6, 7, 8, 9, 10] and Quantum Cryptanalysis of Symmetric and Asymmetric Ciphers [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Improved Quantum Cryptanalysis of AES is given in [22].

In May 2016, IBM launched IBM Q Experience which is web-based platform that gives clients in the overall public access to a set of IBM's model quantum processors via the Cloud. Since then, several modifications and additions have been executed. In most recent set-up, there exist 32 qubits simulator (ibm_qasm_simulator) and several five qubit (ibmqx2, ibmqx4, ibmqx5) and one 16 qubit (16_melbourne) real processors.

We have already implemented a reduced version of quantum A5/1 (10 bits) in IBM Quantum interface using Quantum Information Tool Kit (Qiskit)[23]. In this direction, we achieved the Proof of Concept (PoC) regarding practical implementation of quantum cryptanalysis of stream ciphers. The complete details of about section may refer to [24]. In the following section gives a brief about A5/1 cipher.

A. *Outline of the Paper*

The rest of the paper is organized as follows: we present brief of A5/1 stream cipher, basics of qubits, quantum gates, circuits for random number generator and creating entangled state in section 2. Grover's circuit compose using IBMQ circuit in section 3. Reduced A5/1 cipher in quantum discussed in section 4. Number of Gates and Working qubits Required for full scale implementation in section 5. Finally, section 6 presents the conclusive summary.

II. Preliminaries

In this section we explain the complete explanation of A5/1 cipher adopted in our paper and two main quantum algorithms Deutsch-Jozsa and Grovers[25], which we use to derive our results.

A. *A5/1 Stream Cipher*

In the GSM standard, Stream Cipher A5/1 is used for encrypting transmission. The arrangement of a GSM transmission is a chain of bursts. Any common channel with one direction, each 4.615 milliseconds one burst is sent and consists of 114 bits information. In each burst, keystream of size 114 bits XORed with preceding to modulation of 114 bits. The initialisation of A5/1 uses 64-bit key and fame number F of size 22-bit [26].

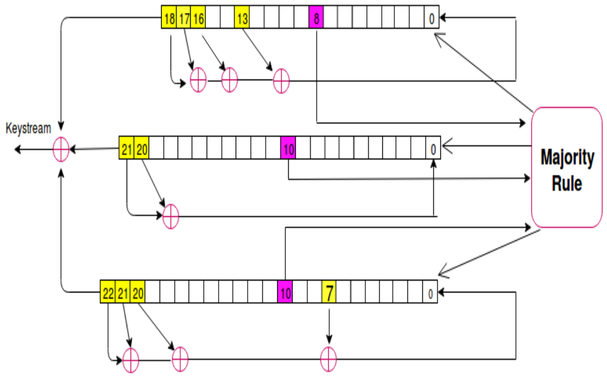


Figure. 1: A5/1 LFSRs

In Figure. 1 LFSR specifications are given. In Table 1 explains the A5/1 [27] stream cipher which is a combination of three linear feedback shift registers (LFSRs). Every register is link with one of the clocking bit. Using majority rule, the registers are clocked.

Length in bits	Feedback Polynomial	Control bit	Tap positions
19	$x^{19}+x^{18}+x^{17}+x^{14}+1$	8	$13, 16, 17, 18$
22	$x^{22}+x^{21}+1$	10	$20, 21$
23	$x^{23}+x^{22}+x^{21}+x^{8}+1$	10	$7, 20, 21, 22$

Table 1: Specifications of A5/1 stream cipher

Using clocking bit, For each and every cycle the majority bit is determined. The register is clocked if majority bit is same as the clocking bit. Two or three registers are clocked for every cycle.

B. *Basic of qubits and the algebra*

In a classical computer the basic elements are 0 or 1 represented as a bit. The bottom line element in quantum criterion is called the qubit or quantum bit. The physical counterpart of a qubit is photon. The representation of qubit is denoted as $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers, satisfies $|\alpha|^2 + |\beta|^2 = 1$. Suppose any one measures the qubit in $\{|0\rangle, |1\rangle\}$ basis, then $|0\rangle, |1\rangle$ measures with probabilities $|\alpha|^2, |\beta|^2$. After qubit observation the initial state gets demolished and break to the observe state. This gives the qubits $|0\rangle, |1\rangle$ are (in quantum) equivalent to classical bits 0 and 1. The qubits $|0\rangle, |1\rangle$ denoted in column vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

Vector $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ represents the superposition of $|0\rangle, |1\rangle$.

Above information gives, each qubit contains boundless information and extract this information is not clear. Further in real quantum implementation of circuits, it is not possible to create a complete qubit with out errors. Finally, conclude that every qubit contains more information than a classical bit.

More than one qubit can be explained as a tensor product in algebra. The tensor product of two qubits represents as follows:

$$(\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle) = \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \otimes \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_1 \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix} \\ \beta_1 \begin{bmatrix} \alpha_2 \\ \beta_2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \alpha_1\alpha_2 \\ \alpha_1\beta_2 \\ \beta_1\alpha_2 \\ \beta_1\beta_2 \end{bmatrix}$$

$$= \alpha_1\alpha_2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_1\beta_2 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \beta_1\alpha_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \beta_1\beta_2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Finally $\alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle$.

All two qubit states can not be expressed as above. Suppose two qubit state $\gamma_1|00\rangle + \gamma_2|11\rangle$ where γ_1, γ_2 be non-zero, can not be expressed as inner product of $(\alpha_1|0\rangle + \beta_1|1\rangle), (\alpha_2|0\rangle + \beta_2|1\rangle)$. This is called as entanglement. Maximally entangled state is called Bell state. Example of Bell state is $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. These bell states are very much importance in quantum information[24].

Quantum gates: The design of a quantum computer essential primitives are quantum gates[24]. Qunatum gate design such a way that equal number of inputs and outputs and these gates are reversable. In scientific this can be viewed as uni-

Input	Quantum gate	Output
$\alpha\lvert 0\rangle+\beta\lvert 1\rangle$	X	$\beta\lvert 0\rangle+\alpha\lvert 1\rangle$
$\alpha\lvert 0\rangle+\beta\lvert 1\rangle$	Z	$\alpha\lvert 0\rangle-\beta\lvert 1\rangle$
$\alpha\lvert 0\rangle+\beta\lvert 1\rangle$	H	$\alpha\frac{\lvert 0\rangle+\lvert 1\rangle}{\sqrt{2}}+\beta\frac{\lvert 0\rangle-\lvert 1\rangle}{\sqrt{2}}$

Table 2: Single input, single output quantum gates

tary matrices of size $2^n \times 2^n$, where these elements are complex numbers. The following Table 2 gives the single input single output quantum gates. These gates in matrix form as follows.

- X gate: $\begin{bmatrix} 0 & \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix}=\begin{bmatrix} \beta \\ \alpha \end{bmatrix}$;
- Z gate: $\begin{bmatrix} 1 & \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix}=\begin{bmatrix} \alpha \\ -\beta \end{bmatrix}$;
- H gate: $\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix}=\begin{bmatrix} \frac{\alpha+\beta}{\sqrt{2}} \\ \frac{\alpha-\beta}{\sqrt{2}} \end{bmatrix}$.

Note that $\frac{\alpha+\beta}{\sqrt{2}}\lvert 0\rangle+\frac{\alpha-\beta}{\sqrt{2}}\lvert 1\rangle=\alpha\frac{\lvert 0\rangle+\lvert 1\rangle}{\sqrt{2}}+\beta\frac{\lvert 0\rangle-\lvert 1\rangle}{\sqrt{2}}$.
In this connection, we now present a simple circuit for true randomness generation using Hadamard gate. One should note that due to the deterministic nature of classical computer, true RNG mechanism is far from realization. Most of the RNGs in use are Pseudo Random Number Generator (PRNG), where a small seed is used as an input and then a deterministic algorithm generates a long stream of random looking data. This is actually not random, as same seed will always generate the same stream of data. Only looking at the data, it may be computationally or information theoretically hard to distinguish the data from a true random source, without knowing the seed. On other hand, in quantum domain True Randomness Generation is possible. The circuit is presented in Figure 2. This circuit can be implemented in IBM quantum computers. The code for the circuit is given in two formats; one is written in qasm editor (Figure 3).

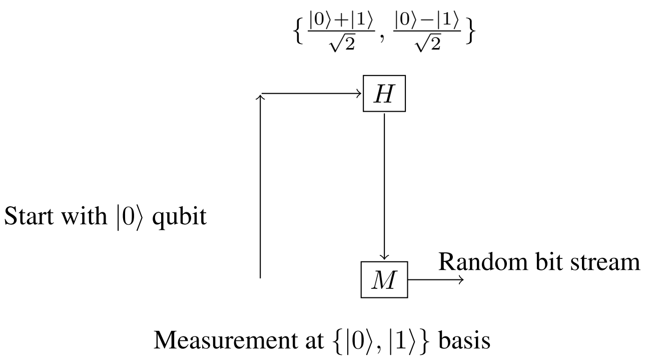
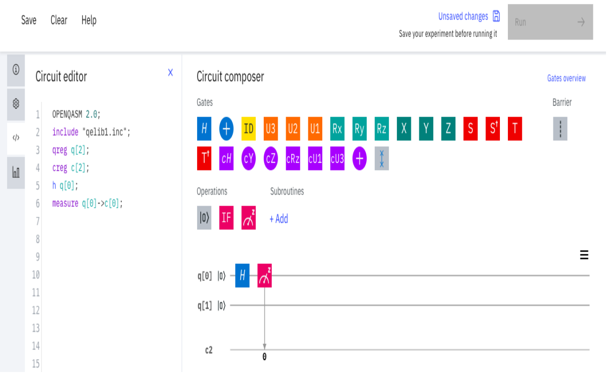

$\{\frac{\lvert 0\rangle+\lvert 1\rangle}{\sqrt{2}},\frac{\lvert 0\rangle-\lvert 1\rangle}{\sqrt{2}}\}$
H
Start with $\lvert 0\rangle$ qubit
M
Random bit stream
Measurement at $\{\lvert 0\rangle,\lvert 1\rangle\}$ basis

Figure. 2: Circuit for True Random Number Generator

We run the code in ibmq/qasm/simulator as well as ibmq/16/melbourne. The histograms are presented in Figure 4 and Figure 5.



```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[2];
creg c[2];
h q[0];
measure q[0]->c[0];
```

Figure. 3: QASM Editor

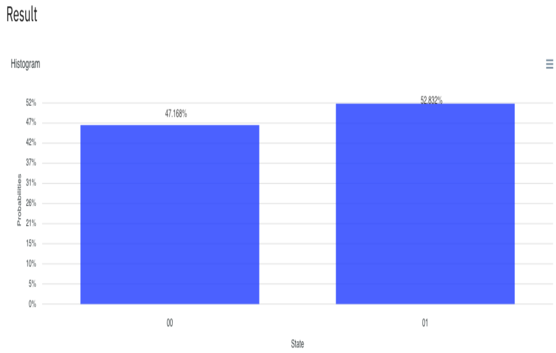


Figure. 4: Simulation

The 4×4 unitary matrices are coming from the 2-i/p, 2-o/p quantum gates. CNOT gate gives the 4×4 unitary matrices is an example as explais below. $\lvert 00\rangle \to \lvert 00\rangle$, $\lvert 01\rangle \to \lvert 01\rangle$, $\lvert 10\rangle \to \lvert 11\rangle$, $\lvert 11\rangle \to \lvert 10\rangle$. The related matrix is $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$. The utilization of these CNOT gate and design of the circuit in Figure 6 to create the entangled states and entangled denoted as: $\lvert\beta_{00}\rangle=\frac{\lvert 00\rangle+\lvert 11\rangle}{\sqrt{2}}$, $\lvert\beta_{01}\rangle=\frac{\lvert 01\rangle+\lvert 10\rangle}{\sqrt{2}}$, $\lvert\beta_{10}\rangle=\frac{\lvert 00\rangle-\lvert 11\rangle}{\sqrt{2}}$, and $\lvert\beta_{11}\rangle=\frac{\lvert 01\rangle-\lvert 10\rangle}{\sqrt{2}}$.
The corresponding qasm program is given in Figure 7. This code has been run in IBMQ simulator and ibmqx2 (actual processor). The histograms are given in Figure 8 and Figure 9. Note that in case of actual processor, error appears in the result.

III. Grover Algorithm in IBMQ

Consider a Boolean function $f(x):\{0,1\}^n \to \{0,1\}$ of size n. Suppose two inputs x, x' are equal then $f(x)=1$ and 0 otherwise. So function representation as follows,

$$\begin{aligned} f(x) &= 1 \quad \text{iff } x = x' \\ &= 0 \quad \text{otherwise} \end{aligned}$$

We now came to know that if we are given a Boolean function $f(x)$, then in quantum domain we always can generate a unitary matrix U_f. Here, we call this U_f as O_f. Now, for O following is the circuit. Consider the following circuit.
The initial state is

$$\lvert\psi_0\rangle=\lvert 0\rangle^{\otimes n}\otimes\lvert 1\rangle$$

First n qubit apply, we can apply the Hadamard gates. Thus

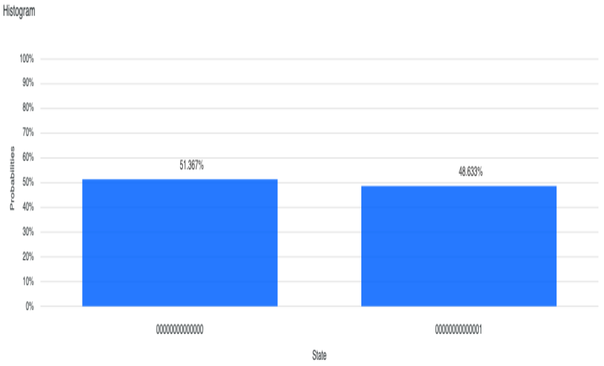


Figure. 5: Actual Processor

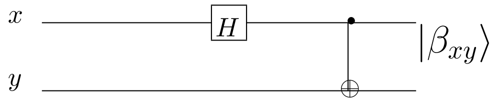


Figure. 6: Quantum circuit for creating entangled state

$|\psi_1\rangle$ becomes

$$|\psi_1\rangle \;=\; \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

We know apply O_f on $|\psi_1\rangle$. We know that for $x \neq x'$ we have,

$$|x\rangle\left(\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle\right)\xrightarrow{O_f}|x\rangle\left(\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle\right).$$

And for $x = x'$ we have,

$$\begin{aligned}|x'\rangle\left(\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle\right)&\xrightarrow{O_f}|x'\rangle\left(\frac{1}{\sqrt{2}}|1\rangle-\frac{1}{\sqrt{2}}|0\rangle\right)\\&=-|x'\rangle\left(\frac{1}{\sqrt{2}}|0\rangle-\frac{1}{\sqrt{2}}|1\rangle\right).\end{aligned}$$

Thus, in general we can write,

$$|x\rangle\xrightarrow{O_f}(-1)^{f(x)}|x\rangle.$$

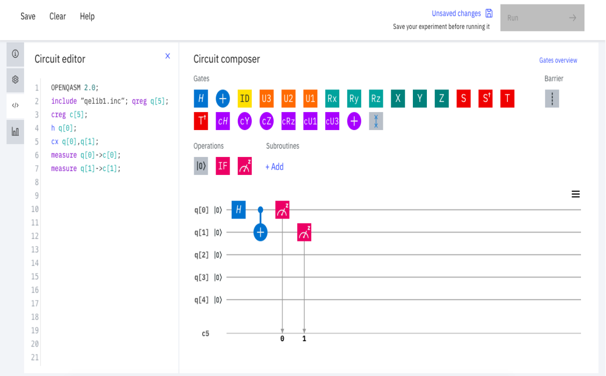
Hence $|\psi_2\rangle$ can be written as

$$|\psi_2\rangle \;=\; \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

If we ignore the state of the last qubit, we can write

$$|\psi_2\rangle \;=\; \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle$$

We now apply a sequence of operations (G)on the output of the oracle O_f. These sequence of operations including the oracle O_f itself is called Grover operation. These sequence of operations includes



```
OPENQASM 2.0;
include "qelib1.inc"; qreg q[5];
creg c[5];
h q[0];
cx q[0],q[1];
measure q[0]->c[0];
measure q[1]->c[1];
```

Figure. 7: QASM program

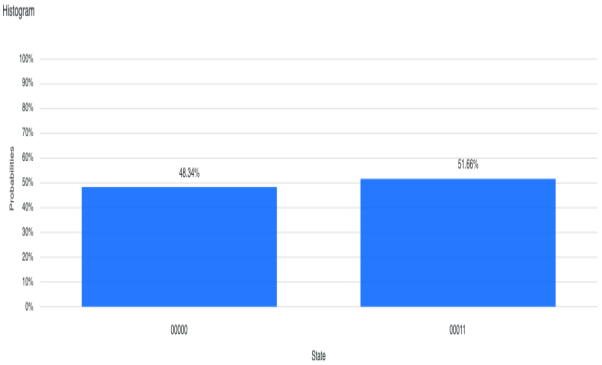


Figure. 8: Simulation; No error exists

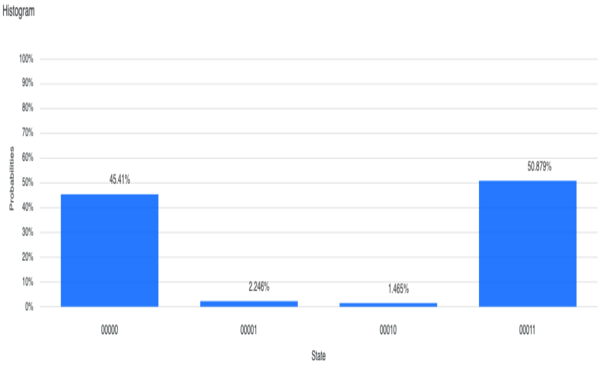


Figure. 9: Actual Processor (ibmqx2): Error appears.

- n number of Hadamard operations
- Conditional phase shift operation P_0
- n number of Hadamard operations

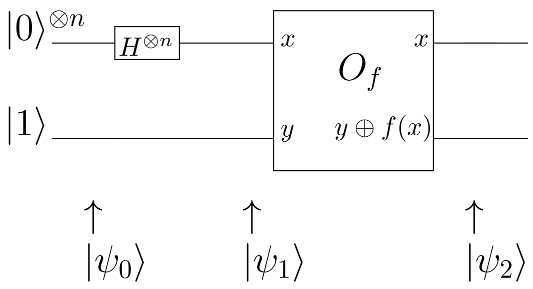
Thus $G = H^{\otimes n}P_0H^{\otimes n}$. Conditional phase shift operator flips the phase of all the conditional basis states except $|0\rangle$. Thus we can write

$$|x\rangle \to (-1)^{\delta_{x,0}}\,|x\rangle$$

Operational representation of this phase shift operator is

$$2\,|0\rangle\,\langle 0| - I,$$

I is the identity matrix.



Now, we will check if this is true. We apply the conditional phase shift operator on a two qubit state $|\phi\rangle = \frac{1}{2}(|00\rangle + |01\rangle\,|10\rangle + |11\rangle)$. Thus, the resultant state $|\phi_1\rangle$ will be

$$\begin{aligned}|\phi_1\rangle &= (2\,|00\rangle\,\langle 00| - I).\frac{1}{2}(|00\rangle + |01\rangle\,|10\rangle + |11\rangle)\\ &= \frac{1}{2}(2\,|00\rangle - |00\rangle - |01\rangle - |10\rangle - |11\rangle)\\ &= \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle - |11\rangle)\end{aligned}$$

Thus, the operational representation of the conditional phase operator is correct.

In Grover's algorithm GO_f is repeated $\frac{\pi}{4}\sqrt{2^n}$ many times. These are called Grover's iterations. After all iterations completed, the first n qubits are measure in the computational basis. The measured state will be the answer.

Let $|x'\rangle$ be the measured answer. Now, set x' as input in the classical circuit for $f(x)$. Check if it gives 1.

A. Analysis of the algorithm

The initial state can be written as,

$$\begin{aligned}|\psi_0\rangle &= \frac{1}{\sqrt{2^n}}|x'\rangle + \frac{1}{\sqrt{2^n}}\sum_{x\neq x_0}|x\rangle\\ &= \frac{1}{\sqrt{2^n}}|x'\rangle + \sqrt{\frac{2^n-1}{2^n}}|\phi_0\rangle\\ &= \sin\theta|x'\rangle + \cos\theta|\phi_0\rangle,\end{aligned}$$

where $\theta = \arcsin\frac{1}{\sqrt{2^n}}$. Note that this can be viewed as a reflection of $|\psi_0\rangle$ about $|\phi_0\rangle$. So $O_f\,|\psi_0\rangle$ is the mirror image of $|\psi_0\rangle$. Here, $|x'\rangle$ plays the role of the mirror.

Now, it can be shown that after the application of $G = H^{\otimes n}P_0H^{\otimes n}$, $O_f\,|\psi_0\rangle$ is again reflected about the axis $|\psi_0\rangle$ and creates an image $GO_f\,|\psi_0\rangle$. (Imagine a mirror which is perpendicular to $|\psi_0\rangle$). As we know that reflected angle will always be equal to the incident angle, here $GO_f\,|\psi_0\rangle$ creates 2θ angle with $|\psi_0\rangle$. Thus, the total angle between $GO_f\,|\psi_0\rangle$ and $|\phi_0\rangle$ becomes 3θ. Hence, we can now say that each application of GO_f amplifies the angle from θ to 3θ.

After k application of amplitude amplification operator GO_f, the resulting state is of the following form,

$$|\psi_k\rangle = \sin(2k+1)\theta|x_0\rangle + \cos(2k+1)\theta|\phi_0\rangle.$$

Probability of observing $|x'\rangle$ from $|\psi_k\rangle$ is $\sin^2(2k+1)\theta$. Thus, The success probability is $\sin^2(2k+1)\theta$. To make the success probability $\frac{1}{2}$ we need,

$$\begin{aligned}\sin^2(2k+1)\theta &= \frac{1}{2}\\ (2k+1)\theta &= \arcsin\frac{1}{\sqrt{2}}\\ (2k+1)\theta &= \frac{\pi}{4}\\ k &\approx \frac{\pi}{8\theta}\\ &= \frac{\pi}{8}\sqrt{2^n}\left[\text{As } \theta\to 0 \implies \sin\theta\approx\theta = \frac{1}{\sqrt{2^n}}\right].\end{aligned}$$

The geometrical diagram of Grover search algorithm has been presented below Figure 10.

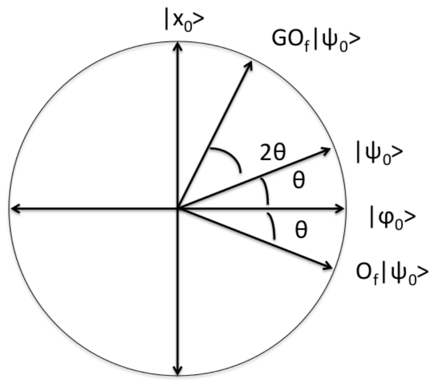
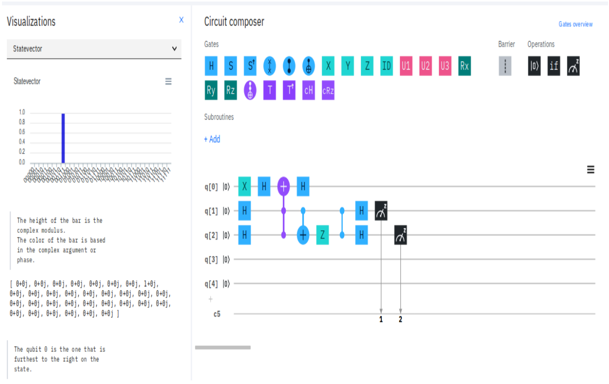


Figure. 10: Geometrical diagram of Grover's algorithm

B. Grover in IBMQ Circuit Compose

Grover Search algorithm for two-input Boolean AND function is presented in Figure 11. Note that after measurement, we are getting all 1 bit string.



The height of the bar is the complex modulus.
The color of the bar is based in the complex argument or phase.

```
[ 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 1+0j,
0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j,
0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j,
0+0j, 0+0j, 0+0j, 0+0j, 0+0j, 0+0j ]
```

The qubit 0 is the one that is furthest to the right on the state.

Figure. 11: Grover Circuit

Some of the symmetric key algorithms are vulnerable in a model that allows superposition attacks. In most pratical circumstances these attacks are not realistic. For example, newly, there have been considerable cryptanalysis results to solve system of non-linear equations for breaking some symmetric algorithms. Solving these non-linear equations is then attacked using a altered version of the quantum linear equation solver algorithm. The results are primarily dependent on the number of the non-linear system, which turns to be difficult to compute. Given the condition number is somewhat little, then, at that point one might get an advantage contrasted with brute-force Grover search.

Theorem 1. *Classical circuit size $T(n)$, If $f : \{0,1\}^n \rightarrow \{0,1\}$ is determinable with above size, then there exists a quantum circuit of size $O(T(n))$. The mapping is $|x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle$, likely $O(T(n))$ extra working bits are used.*

In case of A5/1, the implementation of the theorem is as follows.

Step 1:

1. The function on A5/1 defined as $f : \{0,1\}^n \rightarrow \{0,1\}^n$.
2. In classical domain the circuit requires polynomial many gates and bits.
3. From theorem above, one can construct a quantum circuit $U_f : |x\rangle |y\rangle \rightarrow |x\rangle |y \oplus f(x)\rangle$ with polynomial many quantum gates and qubits. Here, x and y are n-bit strings.

Consider the circuit in Figure 12.

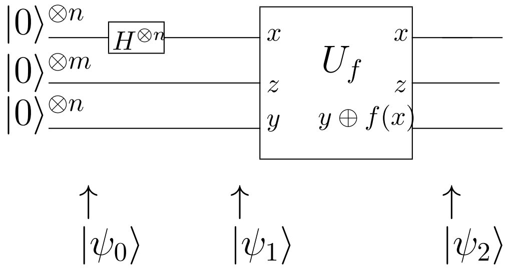


Figure. 12: Circuit for U_f, where f is A5/1 function

Step by step analysis of the circuit is itemized below.

- Start with $2n + m$ many $|0\rangle$. Initial state

$$|\psi\rangle_0 = |0\rangle^{n+m} |0\rangle^n$$

  - First n qubits are used to create equal superposition of all 2^n many possible inputs, i.e.,

$$|x\rangle = \sum_{i \in \{0,1\}^n} |i\rangle$$

  - Next m qubits are used to generate majority voting, i.e., are used for the clocking functionality f'

- Remaining n $|0\rangle$s are exploited to store the output bit stream
- f is followed by f^{-1} so that any input string $|x\rangle$ and $|z\rangle$ remain unaltered
- $|y\rangle$ is set to $|0\rangle$ so that $|y \oplus f(x)\rangle = |f(x)\rangle$
- We can write

$$\begin{aligned}
|\psi_0\rangle &= |0\rangle^n \otimes |0\rangle^m \otimes |0\rangle^n \\
|\psi_1\rangle &= \sum_{i \in \{0,1\}^n} |i\rangle \otimes |0\rangle^m \otimes |0\rangle^n \\
|\psi_2\rangle &= \sum_{i \in \{0,1\}^n} |i\rangle \otimes |0\rangle^m \otimes |f(i)\rangle
\end{aligned}$$

- Thus, $U_f : |x\rangle |z\rangle |y\rangle \rightarrow |x\rangle |z\rangle |f(x)\rangle$

In the next phase, $|\psi\rangle_2$ and one $|0\rangle$ will be the input of the Grover search algorithm. Extra n qubits which represent the keystream $k = k_1 k_2 \cdots k_{64}$ will also be the inputs of Grover algorithm. Grover function (f_G) will be 1 iff $|f(x)\rangle = |k\rangle$ for some x and 0 otherwise, i.e.,

$$\begin{aligned}
f_G(f(x)) =& 1 \quad \text{iff } f(x) = k \\
=& 0 \quad \text{otherwise}
\end{aligned}$$

Grover iteration $G = (2|\psi_2\rangle\langle\psi_2| - I)O$ are repeated for $\sqrt{2^n}$ many times, where O is the Grover oracle. In other words, $O \equiv U_{f_G}$. In $\Omega(\sqrt{2^n})$, we will get $|x\rangle \otimes |z\rangle \otimes |1\rangle^{\otimes n+1} \otimes |k\rangle$ while measure all the registers. $|x\rangle$ will be the secret key. Sometimes we may get false positive. To avoid this situation, after getting the key we should compute the A5/1 function f and check if $f(x) = k$. If not we will repeat the entire procedure once again.

We have implemented the reduced version of U_f (RA5/1) using Quantum Information Tool Kit (Qiskit). The classical description of this RA5/1 and the corresponding quantum codes will be presented in the following subsection.

IV. Cryptanalysis of Reduced A5/1 in quantum

We are implementing quantum attack on 10 bits A5/1 in IBM quantum computer, precisely on 32 qubits qasm simulator. Due to the limitation of qubits in IBM interface, we could not implement the cipher in full scale. However, we will be able to deliver the complete code and to provide the estimation of number of work qubits and quantum gates for full scale implementation.

A. Classical Description of RA5/1:

RA5/1 consists of three LFSRs R_1, R_2 and R_3 of lengths 3, 3 and 4 bits respectively. Every register is restore according to its respective feedback polynomial. The tap positions of the LFSRs correlate to respective primitive polynomials. The tap positions of R_1,R_2,R_3 are 2,1;2,0;3,2 respectively. Depend the values on clocking bits, R_1, R_2 and R_3 are clocked irregularly.

B. Majority Rule:

At each cycle, one of the register is clocked iff the majority of all 3 clocking bits is equal to its clocking bit. At every step at least two or three registers are clocked in Table 3.

C_1	C_2	C_3	Majority	R_1	R_2	R_3
0	0	0	0	✓	✓	✓
1	0	0	0		✓	✓
0	1	0	0	✓		✓
1	1	0	1	✓	✓	
0	0	1	0	✓	✓	
1	0	1	1	✓		✓
0	1	1	1		✓	✓
1	1	1	1	✓	✓	✓

Table 3: Majority Rule Table

C. *Parameters Specification for the Shift Registers:*

Parameters of the shift registers R_1,R_2 and R_3 are specified in Table 4 where as the diagram of RA5/1 is given in Figure 13.

LFSR	**Length in bits**	**Feedback Polynomial**	**Clocking bit**	**Tapped bits**
R_1	3	x^3+x^2+1	1	2,1
R_2	3	x^3+x+1	1	2,0
R_3	4	x^4+x^3+1	2	3,2

Table 4: Parameters of RA5/1 stream cipher

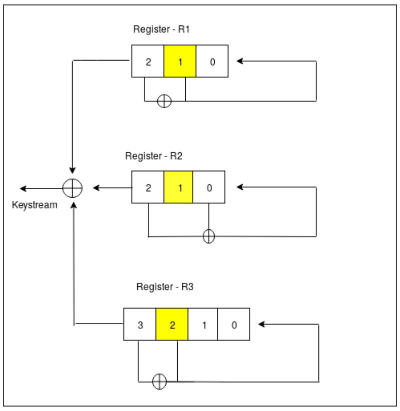


Figure. 13: RA5/1 diagram

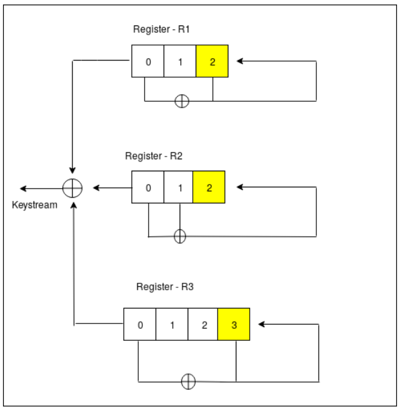


Figure. 14: Reverse RA5/1 diagram

D. *Reverse LFSR specifications:*

Parameters for the shift registers R_1',R_2' and R_3' for reverse RA5/1 are specified in Table 5 where as the schematic diagram is presented in Figure 14.

LFSR	**Length in bits**	**Feedback Polynomial**	**Clocking bit**	**Tapped bits**
R_1	3	x^3+x+1	2	2,0
R_2	3	x^3+x^2+1	2	1,0
R_3	4	x^4+x+1	3	3,0

Table 5: Parameters of Reverse RA5/1

E. *Majority Rule for RU$_f$:*

In quantum domain for the majority rule, we have to define a function. We store the input states of LFSR R_1, R_2 and R_3 in quantum registers q_0, q_1, q_2, q_4, q_5, q_6 and q_8, q_9, q_{10}, q_{11} respectively. Similarly, the output bits from R_1, R_2 and R_3 are stored in q_3, q_7 and q_{12} respectively. Now, we define the function $f : \{0, 1\}^3 \to \{0, 1\}$ as follows.

$$
\begin{aligned}
f \quad = \quad & (\overline{q_1 \oplus q_5}) \wedge (\overline{q_1 \oplus q_{10}}) \wedge (q_3 \oplus q_7 \oplus q_{12}) \\
& \oplus(\overline{q_1 \oplus q_5}) \wedge (q_1 \oplus q_{10}) \wedge (q_3 \oplus q_7) \\
& \oplus(q_1 \oplus q_5) \wedge \overline{(q_1 \oplus q_{10})} \wedge (q_3 \oplus q_{12}) \\
& \oplus\overline{(q_5 \oplus q_1 0)} \wedge \overline{(q_1 \oplus q_{10})} \wedge (q_7 \oplus q_{12})
\end{aligned}
$$

F. *Number of Gates and Work Qubits Required for RA5/1*

Number of gates and work qubits required for RA5/1 is tabulated in Table 6.

S.No	**Name of the Gate**	**Number of Gates**
1	H	10
2	CNOT	23
3	X	3
4	SWAP	7
5	Toffoli	7
6	Number of work qubits	27

Table 6: Number of gates required for RA5/1

The following Figure 15 gives the complete structure of RA5/1

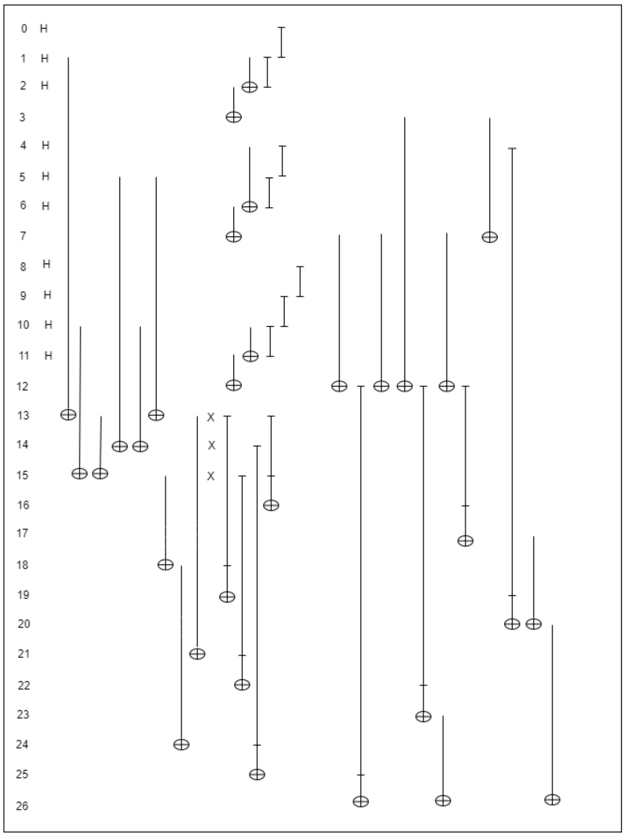


Figure. 15: complete structure of RA5/1

G. Qiskit Program for RA5/1

Qiskit program for Reduced A5/1 is given below

```
######### initial state #########
qc.h(q[0])
qc.h(q[1])
qc.h(q[2])
qc.h(q[4])
qc.h(q[5])
qc.h(q[6])
qc.h(q[8])
qc.h(q[9])
qc.h(q[10])
qc.h(q[11])
######### clocking bits #########
qc.cx(q[1],q[13])
qc.cx(q[10],q[15])
qc.cx(q[13],q[15])
qc.cx(q[5],q[14])
qc.cx(q[10],q[14])
qc.cx(q[5],q[13])
qc.cx(q[18],q[24])
qc.cx(q[13],q[21])
qc.x(q[13])
qc.x(q[14])
qc.x(q[15])
######### LFSR−1 #########
qc.cx(q[2],q[3])
qc.cx(q[1],q[2])
qc.swap(q[1],q[2])
qc.swap(q[0],q[1])
######### LFSR−2 #########
qc.cx(q[6],q[7])
qc.cx(q[4],q[6])
qc.swap(q[5],q[6])
qc.swap(q[4],q[5])
######### LFSR−3 #########
qc.cx(q[11],q[12])
qc.cx(q[10],q[11])
qc.swap(q[10],q[11])
qc.swap(q[9],q[10])
qc.swap(q[8],q[9])
qc.ccx(q[13],q[18],q[19])
qc.ccx(q[15],q[21],q[22])
qc.ccx(q[14],q[24],q[25])
qc.ccx(q[13],q[14],q[16])
qc.cx(q[7],q[12])
qc.ccx(q[12],q[25],q[26])
qc.cx(q[7],q[12])
qc.cx(q[3],q[12])
qc.ccx(q[12],q[22],q[23])
qc.cx(q[7],q[12])
qc.ccx(q[12],q[16],q[17])
qc.cx(q[3],q[7])
qc.ccx(q[7],q[19],q[20])
##########################
qc.cx(q[23],q[26]) # 6,8 output
qc.cx(q[17],q[20]) # 2,4 output
qc.cx(q[20],q[26]) # final output
```

Output of RA5/1 in Table 7. Final output stored in q_{26}

1000001001111101001011111100:10	0110010010001000110001110101:10	0110010010001011100111101011:9
0010000110000101001000000000:10	1000001001111101001011111100:10	0110010010001000110001110101:10
0110010010001011100111101011:9	0010000110000101001000000000:10	0000110000001001011111011001:4
0010000110000101011110101011:7	0000000000111100100000000010:9	0110010010001000110000000101:9
1000001001111111100111101100:7	1010001110000100110110101101:7	0000000000111110111000001011:7
1010001110000110001011111100:12	0000110000001000010001110010:6	1110010010001010001001111110:10
0010000110000110111010000101:13	1001110000001001100010100101:5	1010001110000110101111010101:9
1000001001111111111011101100:6	1000001001111100100000001001:5	0000000000111111000001001100:10
1110010010001001011111100010:9	0010000110000111100101010111:9	0000110000001001010111101001:4
1001110000001001010101110000:9	0010000110000110001110111100:11	1001110000001000110100111110:5
1110010010001010001000011100:7	0000110000001000101100110101:9	0000000000111111110001111100:11
1001110000001001011111100000:9	0000110000001000001011010010:5	0010000110000111000101011001:6
1110010010001001111111011011:7	0000000000111110001101000000:6	1000001001111110101101011011:11
1000001001111110101101011001:14	0000110000001000100100000101:7	1110010010001000010101011110:7
0000110000001000100110100111:10	1010001110000100100011111110:11	0110010010001011110110100010:6
1000001001111100010101000010:10	0000110000001000010011010010:2	1000001001111100010100100000:6
0000110000001000101110010101:8	0110010010001000000000110111:9	1000001001111111101011101100:6
0000110000001001011111101001:3	1001110000001000110100111100:14	0000000000111100100101010111:7
1000001001111110101000000010:5	0000110000001001110010011110:13	0110010010001001001010010111:12
1000001001111110011000000000:7	0000110000001000101110100101:9	0000110000001000111110010111:7
0000110000001001111000111110:14	0110010010001000000000110101:9	0000000000111111101001111110:7
0010000110000110111111011110:7	0110010010001010111110010101:4	0000110000001001110010101100:6
1010001110000110011111010101:6	1110010010001001011111101011:6	0000000000111111111001001110:8
0000000000111110001000001011:9	0110010010001000100000001110:8	1000001001111101011111100111:11
1001110000001001111000110101:6	0000110000001000011001000000:8	0000110000001000100110010101:6
0010000110000100010010000101:7	0000110000001000110100110101:6	0000110000001001100000001100:5
1010001110000101101000001001:16	1010001110000111000000111011:6	1001110000001001000111010010:6
0010000110000101111110101001:6	1010001110000111000000001001:9	0000000000111111110001001100:6
0000000000111100000010101001:7	1110010010001000100100111110:13	1110010010001001101111011011:11
0110010010001011010110111011:10	0010000110000101101000000000:7	0110010010001011110111011001:10
0110010010001011000111101011:10	1000001001111110111110101011:9	1110010010001000110101011100:9
0010000110000111111010011001:8	0000000000111111000111001111:6	0010000110000111000100111001:13
0000110000001000000001000000:7	0110010010001010111100111100:9	1010001110000110111101001110:7
0010000110000100000001110101:8	1000001001111111110011101110:6	1010001110000100010011111110:7
1110010010001000010101011100:6	0010000110000100110111011110:6	1110010010001010101010000111:7
0110010010001000000000001100:4	0110010010001011000110100000:9	0000000000111100110001100000:10
1110010010001000010101001111:10	1010001110000100100011111100:5	0000000000111100100000000000:6
1001110000001000100110101110:7	1010001110000110101010011100:8	0010000110000101101101010111:12
0000000000111111010010011100:7	1001110000001000011011011001:8	0000000000111110101101000010:8
0000000000111111110010011100:9	1000001001111100010100100010:8	1010001110000110001011111110:4
0010000110000111000100111011:7	0010000110000110101010000111:9	1000001001111101011111100101:7
1010001110000111010101000000:8	1010001110000101101001111011:11	0010000110000100100010000111:10
0010000110000100000001110111:11	0110010010001000100001111110:8	0000000000111011111011110111:7
0000000000111111100111001111:5	0110010010001010011101011100:7	1001110000001001010111100000:9
0000110000001001000111011011:8	1110010010001001101111010010:7	0000000000111100110101011001:6
1001110000001000000100101011:8	1001110000001000100100001100:6	0000000000111110111101000000:13
0010000110000111110001100000:10	0110010010001010001110101100:6	0000110000001000010011100000:5
1000001001111111111001110101:10	0000110000001001111000001110:9	1001110000001001100010100111:8
1110010010001000110101001111:11	0000110000001000110100110111:6	1110010010001011000001111011:6
0000000000111111000001001110:10	1001110000001001011111010000:7	1110010010001011000001110000:12
1110010010001000100100100101:8	1001110000001000001001111001:3	1010001110000101001100100000:4
1010001110000110111111100111:12	0110010010001011110111101001:13	1110010010001001101111100000:2
0000000000111100110000001011:5	0000110000001000000001110010:7	0000000000111100010001100010:11
0000000000111100111010000000:8	0000000000111111010011111100:11	0000000000111110011001111001:6
1001110000001001110000000111:7	0010000110000111010000000000:8	1010001110000100110010011100:10
1010001110000111110010010010:6	1010001110000111010001111001:9	1010001110000101011001111011:9
0000110000001000100100110101:10	1010001110000111100001111001:10	0110010010001001011011111001:7
1010001110000101111101000010:9	1000001001111100110001111011:5	0110010010001010001100111110:8
1000001001111110101001100000:5	1010001110000101101001111001:9	0000110000001001101010101100:7
1000001001111100100101000000:7	1000001001111100110101000000:8	1010001110000101111000001011:7
0010000110000111010100111001:9	0110010010001001111101000000:13	1000001001111100000000001001:11
1001110000001001000101000000:8	0010000110000101011000000010:5	0000110000001001101010011110:9
0110010010001011010111101011:13	0000110000001001101000111100:9	1010001110000111000101000000:11
0000110000001001101010011100:18	1110010010001010001001111100:5	1110010010001001111111100000:10
0000110000001000111100110101:5	0010000110000100000011011110:8	1000001001111101011111010111:8
0000110000001000111100000101:6	0000000000111110001100100000:5	1110010010001000000010100101:8
0110010010001001011011111011:4	0010000110000111110101010111:3	1110010010001000010101000101:6
1110010010001000100101011110:7	0000000000111101011011110101:8	0010000110000110011010000101:7
0000000000111111010110101111:10	0110010010001011100110111001:7	0000110000001000011011010000:8
1010001110000101001000001001:9	0110010010001000010000001100:7	0000110000001001010111011001:10
1110010010001000000010100111:11	0000000000111110101100100000:3	1000001001111101011010011100:7
0010000110000100100010000101:5	1000001001111111000110011001:7	0000000000111100110000000000:8
0000000000111110111100100010:7	1001110000001001111010010101:2	0000110000001000000001110010:8
0000000000111011111111101110:9	0110010010001011010110100010:3	1110010010001010101000011110:4
1001110000001001100010010101:11	0110010010001010101101001111:6	0000000000111101001011110101:4
1001110000001001100000000111:2	0000110000001001010101111001:8	0000110000001001011111011011:7
1010001110000111110100100010:6	0010000110000110011011101111:4	0110010010001000100000000111:10
0110010010001000010000000111:9	0000000000111100010000000000:6	0110010010001010001101011110:4
0110010010001011010111010000:6	1010001110000111110101000000:7	1000001001111100110000001011:8
0000110000001000011001000010:5	0000000000111111111011010101:6	0000110000001000101100000111:7
0000110000001001110000111110:6	0110010010001000100000000101:6	1000001001111101011011111100:11
1010001110000100000111001111:10	0000110000001000010011100010:11	0010000110000110001111101100:6
1001110000001001000101000010:7	1000001001111100110001111001:12	0110010010001010111110100111:8
0010000110000100000011101110:5	0000110000001001001101111011:6	1010001110000101101101000000:10
0010000110000110111111011100:7	1001110000001001000111100010:2	1001110000001001000101110000:5
0000000000111101101011110111:6	1000001001111111010110011001:7	0000000000111100100101011001:8
1110010010001011010011100010:3	0110010010001000000000000101:3	0110010010001000100001111100:10
0010000110000111111010011011:10	1001110000001001010111100010:5	1001110000001001011111100010:7
1010001110000110011111010111:3	1110010010001000010100101111:6	0000110000001000000011010010:11
0000110000001000100110010111:8	0000110000001000111100110111:9	1010001110000111000000001011:6
1001110000001000011001111011:4	0000000000111110101001111011:11	0000110000001001000111101011:10
1010001110000101101100100010:7	0010000110000101001001100000:7	1000001001111101101111010101:2
1110010010001011000001000010:8	1000001001111110000010100000:6	0010000110000110001011110111:5
1001110000001000001011101011:7	1010001110000101101101000010:9	0010000110000111010100111011:8
1000001001111111110001110111:4	0000000000111101001010000101:13	1010001110000101011100100000:6
0010000110000110011111011100:11	1110010010001000000010011110:10	0000000000111111000111100101:5

Table 7: Final keystream stored in q_{26} and number of shots requried for each key stream generation

V. *Estimation*: Number of Gates and Work Qubits Required for complete A5/1

Number of gates and work qubits required for complete A5/1 is tabulated in Table 8.

Name of the Gate	Number of Gates
H	64
CNOT	24
X	3
SWAP	61
Toffoli	8
Number of work qubits	81

Table 8: Number of gates required for A5/1

Three LFSR's used in A5/1 are
LFSR-1: $x^{19} + x^{18} + x^{17} + x^{14} + 1$,
LFSR-2: $x^{22} + x^{21} + 1$,
LFSR-3: $x^{23} + x^{22} + x^{21} + x^{8} + 1$.
The number of gates and qubits required in Table 9.

Name of the Gate	LFSR-1	LFSR-2	LFSR-3
H	19	22	23
CNOT	3	1	3
X	-	-	-
SWAP	18	21	22
Toffoli	-	-	-
Number of working qubits	20	23	24

Table 9: Number of gates required for LFSR's

VI. Conclusion and Future Research Plan

The total number of qubits are required for complete A5/1, together with the required number of Toffoli, swap, cnot and working qbits is summarized in Table 8,9. Our attack model was based on a brute force searching via a parallelized version of Grover's algorithm. Poof of Concept (PoC) of quantum cryptanalysis on A5/1 also exists. We are using IBM quantum interface only. However, quantum preparedness for other interface like Google, Intel, Rigetti etc., became important in the state-of-the-art situation. So effort should be taken in this regard and would be our future researach direction.

References

[1] Feynman, R.P. Simulating physics with computers. Int J Theor Phys 21, 467-488 (1982).

[2] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. Proceedings of Royal Society of London, A439:553-558 (1992).

[3] L. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of 28th Annual Symposium on the Theory of Computing (STOC)*, pages 212-219, May 1996.

[4] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. Foundations of Computer Science (FOCS), page 124-134, IEEE Computer Society Press, 1994.

[5] NSA statement, US National Security Agency, Cryptography today, August 2015, archived on 23 November 2015.

[6] C. H. Bennett, G. Brassard, Quantum cryptography: Public key distribution and coin tossing, In Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, 175, 8, 1984.

[7] C. H. Bennett, Quantum cryptography using any two nonorthogonal states, *Phys. Rev. Lett.*, 68, 3121, 1992.

[8] A. K. Ekert, Quantum Cryptography Based on Bell's Theorem, *Phys. Rev. Lett.*, 67, 661, 1991.

[9] H-K Lo, M. Curty, B. Qi, Measurement-device-independent quantum key distribution, *Phy. Rev. Lett.*, 108, 130503, 2012.

[10] V. Scarani, A. Acín, G. Ribordy, N. Gisin, Quantum Cryptography Protocols Robust against Photon Number Splitting Attacks for Weak Laser Pulse Implementations *Phys. Rev. Lett.*, 92, 057901, 2004.

[11] H. Kuwakado, M. Morii, Quantum distinguisher between the 3-round feistel cipher and the random permutation. In: IEEE International Symposium on Information Theory, ISIT 2010, June 13-18, 2010.

[12] H. Kuwakado, M. Morii, Security on the quantum-type even-mansour cipher. In: Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012, Honolulu, HI, USA, October 28-31,, IEEE (2012) 312-316, 2012.

[13] M. Kaplan, G. Leurent, A. Leverrier, M. Naya-Plasencia, Breaking symmetric cryptosystems using quantum period finding, Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Volume 9815 of Lecture Notes in Computer Science., Springer (2016).

[14] G. Leander and A. May, Advances in Cryptology, Grover Meets Simon, Quantumly Attacking the FX-construction, pp 161-178, ASIACRYPT 2017.

[15] A. Chailloux, M. Naya-Plasencia, and A. Schrottenloher, An Efficient Quantum Collision Search Algorithm and Implications on Symmetric Cryptography ASIACRYPT 2017: Advances in Cryptology, pp 211-240, ASIACRYPT 2017.

[16] Suo, J., Wang, L., Yang, S. et al. Quantum algorithms for typical hard problems: a perspective of cryptanalysis. Quantum Inf Process 19, 178 (2020).

[17] G. Alagic and A. Russell, Quantum-Secure Symmetric-Key Cryptography Based on Hidden Shifts EUROCRYPT 2017: Advances in Cryptology, pp 65-93, EUROCRYPT 2017.

[18] X. Bonnetain and M. Naya-Plasencia, Hidden Shift Quantum Cryptanalysis and Implications, ASIACRYPT 2018: Advances in Cryptology, pp 560-592, ASIACRYPT 2018.

[19] Kaplan, M., Leurent, G., Leverrier, A., & Naya-Plasencia, M. (2016). Quantum Differential and Linear Cryptanalysis. IACR Transactions on Symmetric Cryptology, 2016.

[20] B. Langenberg, H. Pham and R. Steinwandt, ”Reducing the Cost of Implementing the Advanced Encryption Standard as a Quantum Circuit,” in IEEE Transactions on Quantum Engineering, vol. 1, pp. 1-12, 2020.

[21] Anand, R., Maitra, S., Maitra, A., Mukherjee, C.S., Mukhopadhyay, S.: Resource estimation of grovers-kind quantum cryptanalysis against fsr based symmetric ci- phers. Cryptology ePrint Archive, Report 2020/1438 (2020).

[22] B. Langenberg, H. Pham and R. Steinwandt, ”Reducing the Cost of Implementing the Advanced Encryption Standard as a Quantum Circuit” in IEEE Transactions on Quantum Engineering, vol. 1, pp. 1-12, 2020.

[23] QISKIT. https://qiskit.org/.

[24] M. A. Nielsen and I. L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2010.

[25] S. P. Jordan and Y.K. Liu, ”Quantum cryptanalysis: Shor, grover, and beyond” IEEE Secur. Privacy, vol. 16, no. 5, pp. 14-21, Sep. 2018.

[26] Jovan Golic, Cryptanalysis of Alleged A5 Stream Cipher, Proceedings of Eurocrypt ’97, Springer LNCS vol. 1233, pp. 239-255, 1997.

[27] Marc Briceno., Ian Goldberg., David Wagner.: A pedagogical implementation of the GSM A5/1 and A5/2 airvoice privacy encryption algorithms, 1999.

Author Biographies

Swamy Naidu Allu is pursuing his Ph.D in computer science & engineering, university college of science from Acharya Nagarjuna University, Guntur, Andhra Pradesh and also working as Research Associate in CR Rao Advanced Institute of Mathematics, Statistics, and Computer Science (AIMSCS), University of Hyderabad Campus, Hyderabad. He did his M.Tech in Computer Science and Data Processing from IIT Kharagpur. His areas of interest are Cryptography and Cryptanalysis, Quantum and Post Quantum Cryptography, Coding theory.

Appala Naidu Tentu is an Associate Professor at CR Rao Advanced Institute of Mathematics, Statistics, and Computer Science (AIMSCS), University of Hyderabad Campus, India. He obtained Ph.D in Computer Science and Engineering (specialisation in Cryptography and Information Security) from JNTU Hyderabad and M.Tech in Computer Science from National Institute of Technology, Surathkal, Karnataka. His research interests are in the areas of cryptography, cryptanalysis and design of security protocols.