MINOR-1 PROJECT REPORT

on

ALGORITHM FOR IMPROVING SECURITY IN PUBLIC CLOUDS USING CIRCULAR QUEUES AND FIBONACCI NUMBERS

Submitted By:

Name	Roll No	Branch	Sap-id
Apaar Sharma	R110216032	CSE CCVT	500052272
Devanshu Dwivedi	R110216059	CSE CCVT	500053470
Sarthak Kathuria	R110216137	CSE CCVT	500052058

Under the guidance of

Nitin Arora
Assistant Professor (SS)
Department of Computer Science

School of Computer Science
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
Dehradun-248007, 2018-2019

Approved By

(Mr. Nitin Arora)
Project Guide

(Dr. Amit Agarwal)
Department Head

School of Computer Science
University of Petroleum & Energy Studies, Dehradun

Synopsis Report (2018-19)

1. Project Title

ALGORITHM FOR IMPROVING SECURITY IN PUBLIC CLOUDS USING CIRCULAR QUEUES AND FIBONACCI NUMBERS

2. Abstract

Security of data in public clouds is one of the most critical challenge for developers and hackers. To preserve data integrity, there is a great need for more reliable security technologies.
This is a low complexity circular queue data structure-based algorithm. Use of Multiple complicating variable factors is the strength of this encryption/decryption algorithm which makes the recovery of stolen data very difficult for the attacker. These variable factors are the size of the circular queue, the beginning of the chosen keyword and the multiple representations in the Fibonacci format.
Firstly, all letters are converted into their ASCII binary form so that they can be used by the proposed security algorithm as it uses shift and logical XOR operations.
Circular queue is used as the time complexity for deQueue and enQueue operations is very fast that is O(1). The variable challenging factor provide flexibility in changing the security of the algorithm as per the circumstances.

Keywords: Data Encryption, Public Cloud, Data integrity, circular queue, Fibonacci representations, variable complexity factors.

3. Introduction

Encryption of data for its authenticity and consistency has become a vital part in the technological era. This has become such an important issue due to several modern concerns which include different kind of cyber-attacks, privacy breaches, boom in the cloud computing area and the increase in the amount of electronic transactions.
Many tools for securing the data over the internet are already present like cryptography, steganography, water marking and different algorithms used to prevent data integrity.
By implementing these security mechanisms, enterprises can more readily identify data breaches and data tampering if it occurs and the data is less likely to be leaked or hacked or at least it is secure even if it's stolen.
This algorithm deals with the use of encryption and decryption to protect data from unauthorized use in public clouds. The model proposed uses circular queue and multiple variable complicating factors to protect the data from the attack of hackers and other unauthorized personal.

4 Literature review

Circular queue is a very efficient data structure as the time complexity for the deQueue and enQueue operation is just O(1). Circular queue can also be employed to enhance data security by making the process of ciphering and deciphering data much more difficult.

The authors of paper [1] developed different techniques like shifting and replacing operations of bi-column-bi-row have already been implemented to increase data security in the past. A random number was used in this algorithm for the shifting process. This led to an increase in the complexity of the decryption of the ciphered text. A similar algorithm was designed by the author of [2] in which matrix scrambling was implemented using circular queue.

Different to the traditional method, the author of [3] proposed a double encryption double decryption algorithm, in this the transmitter encrypts the text two times and the receiver decrypts it using a public key. Also an algorithm was proposed in which the text is firstly converted into ASCII code and after that prime numbers and random numbers are chosen to form binary format[4].

Previously, Fibonacci sequence was mostly used for image encryption. In [5] an algorithm was developed which converted text to Fibonacci sequence, now this Unicode was converted to hexadecimal to form a RGB matrix. Now this was used to finally form the image.

5 Problem statement

In today's world, everything is on the internet. With this paradigm shift and boom in public clouds it has become necessary to protect this data from the attacks of hackers and maintain the Data Integrity. So now the problem at hand is to develop a data encryption/decryption algorithm which can provide data security, does not highly impact the time taken to access the data and provide flexible security measures which can be changed at any time as agreed upon by the sender and receiver.

To become successful in todays competitive market of public clouds it is necessary to earn the customers trust and make data available to him at all time without any substantial latency. Thus, the encryption/decryption of the data should be not only secure but fast as well.

6 Objectives

The primary objective is to come up with a secure encryption/decryption algorithm to prevent data on the cloud from getting into unauthorized hands. The model proposed uses circular queue and multiple variable complicating factors to protect the data from the attack of hackers and other unauthorized personal.

The only two fears preventing people from moving to cloud are security and availability of data at all time. Hence, our objective is to make the retrieval of the data fast and with minimum latency. The data should be available at all time, so the algorithm should work in real time.

7 System Requirements

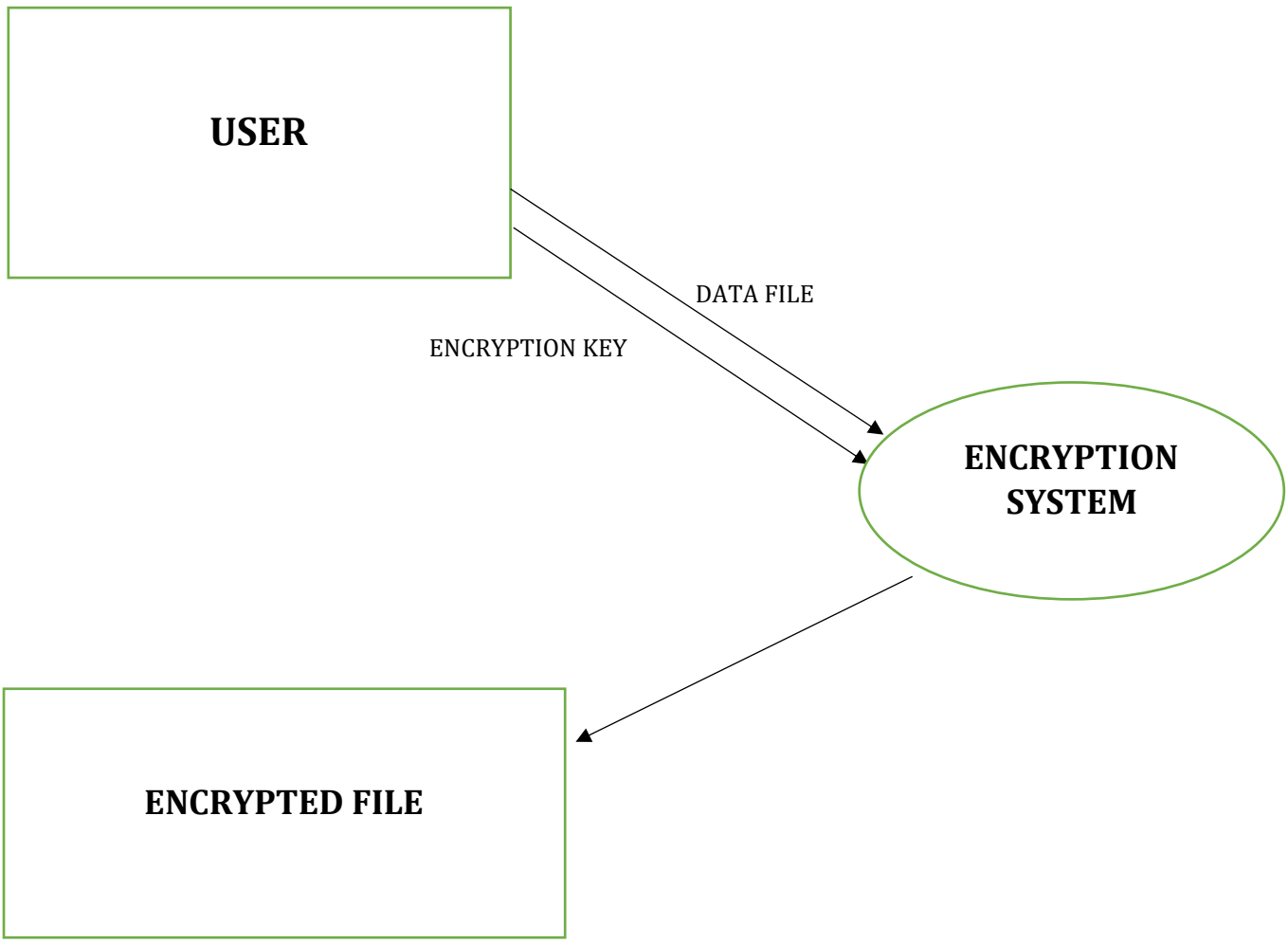
Hardware:

1) x86 architecture
2) cloud hardware components

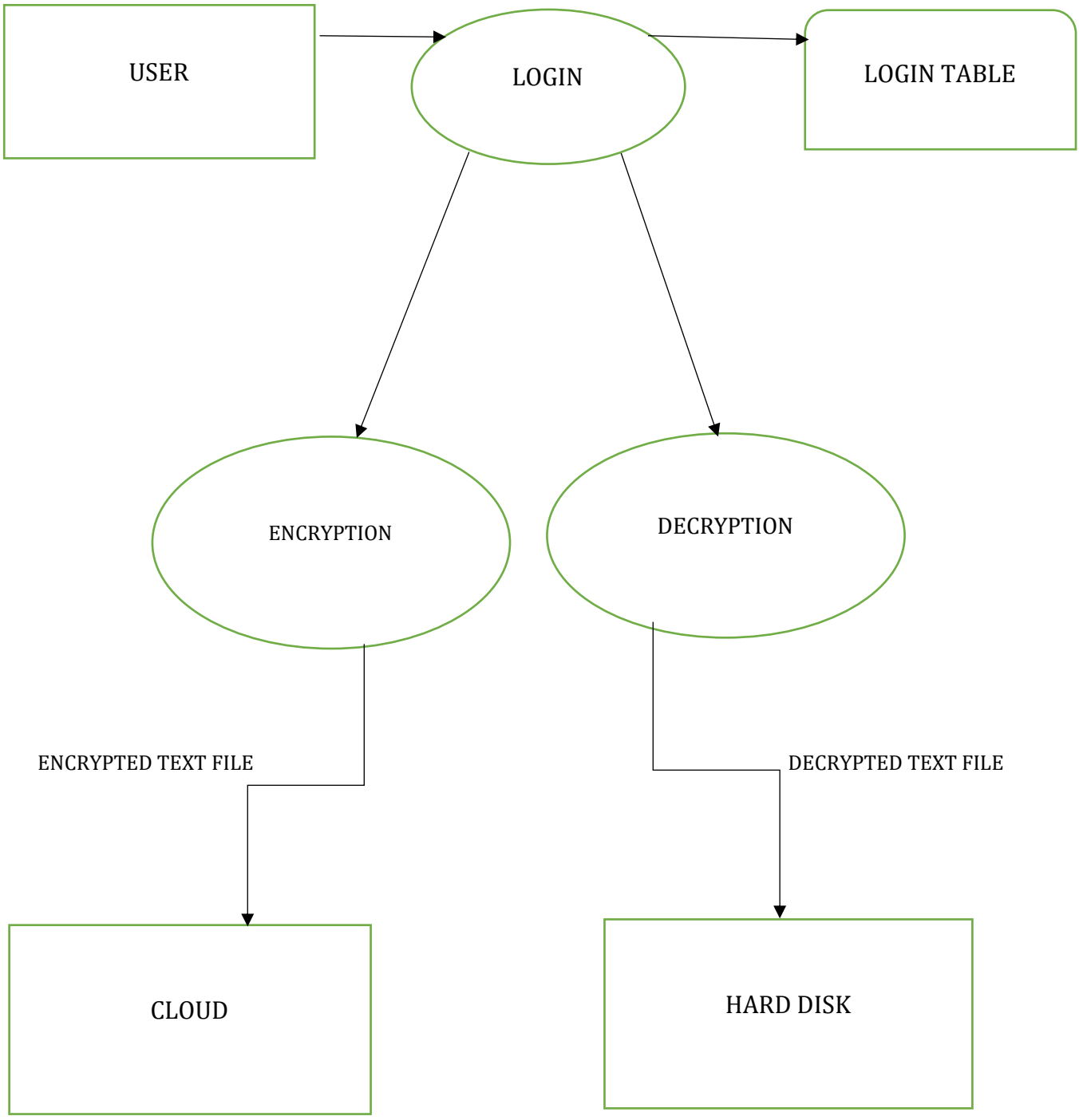
Software:

1) GCC GNU Compiler
2) VMware
3) IDE
4) Microsoft azure
5) puTTy

8 DATA FLOW DIAGRAM (LEVEL 0)

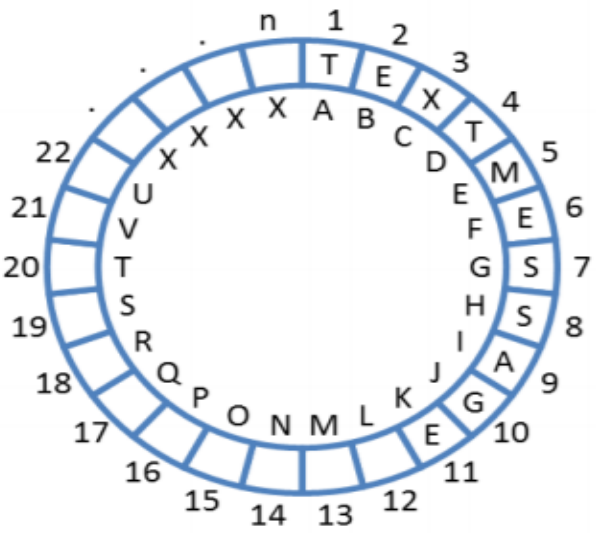


9 DATA FLOW DIAGRAM (LEVEL 1)



10 Methodology

This encryption and decryption process make use of Circular queue data structure.



In the picture given below, ‘n’ represents the size of the circular queue and the plaintext to be encrypted is shown inside the queue. The letters inside the circle represents the keywords that are to be shifted to the right and left and then XORed with plaintext to obtain the cipher text.

Using circular queue provides us with several factors which make the process of encryption/decryption much more difficult for unauthorized access of data.

The factors agreed upon by both sender and receiver are as follows:

1) The size of the circular queue.
2) The beginning of the keyword letter.
3) The representation number in the Fibonacci format.

The process is divided into the encryption process and the decryption process

A) The encryption processes

Step 1: the sender and receiver agree upon the three factors.

1) The size of the queue
2) The keyword letters
3) The representation of the Fibonacci number used.

Step 2: The encryption process begins by distributing plaintext inside the circular queue.

Step 3: the plaintext letters are XORed with a keyword letter.

For example:

Plain letter	A	0	1	0	0	0	0	0	1
Keyword letter	E	0	1	0	0	0	1	0	1
Encrypted letter		0	0	0	0	0	1	0	0

Step 4: represent the encrypted output as decimal numbers.

For example:

ENCRYPTED LETTER								DECIMAL NUMBER
0	0	0	0	0	1	0	0	4

Step 5: convert the obtained decimal number into Fibonacci format to be sent as binary numbers.

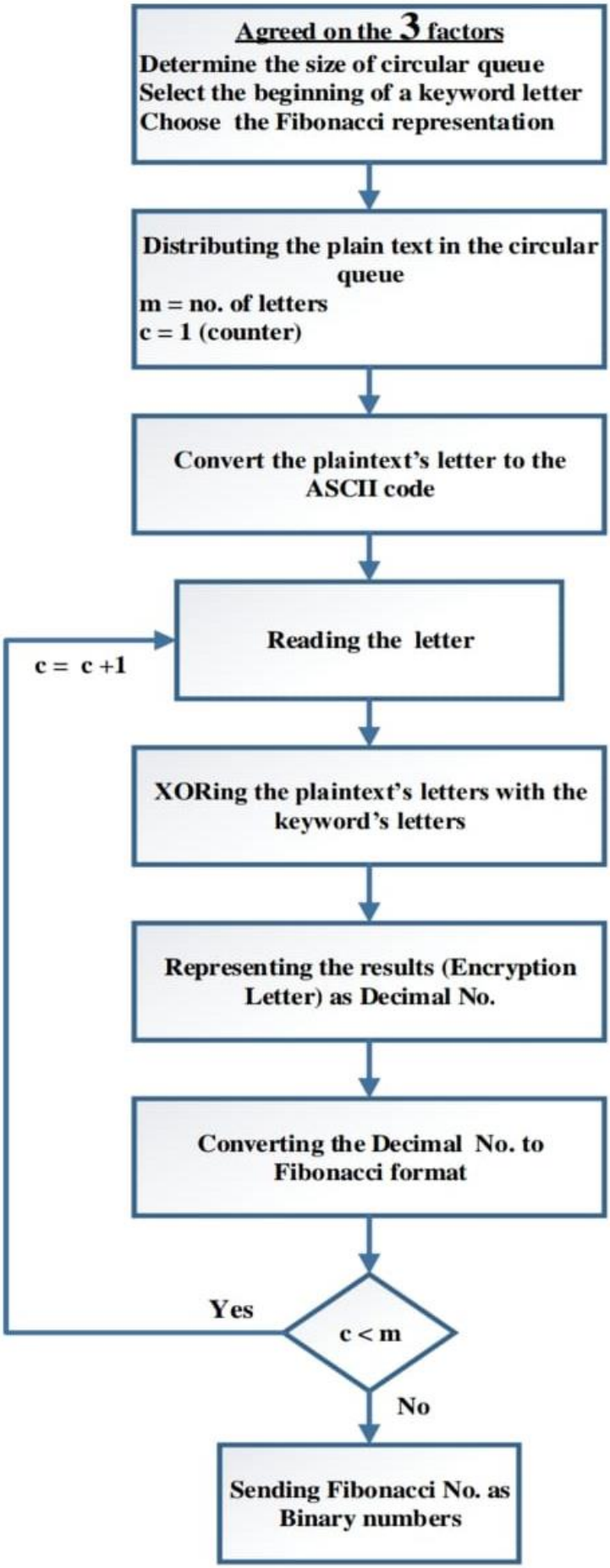
For example:

DECIMAL NUMBER	FIBONACCI FORMAT							
	21	13	8	5	3	2	1	1
4	0	0	0	0	1	0	0	1

Step 6: the final step is to send the Fibonacci numbers as binary numbers that are represented in decimal

For example:

FIBONACCI NUMBERS AS BINARY NUMBERS								DECIMAL NUMBER
0	0	0	0	1	0	0	1	9



ENCRYPTION PROOCESS

2) Decryption process

Step 1: convert the received encrypted message (Fibonacci number) to decimal number

Example:

FIBONACCI NUMBER								DECIMAL NUMBER
21	13	8	5	3	2	1	1	
0	0	0	0	1	0	0	1	4

Step 2: convert decimal number to binary format.

Example:

DECIMAL NUMBER	BINARY FORMAT							
4	0	0	0	0	0	1	0	0

Step 3: XOR binary numbers with the keyword letters.

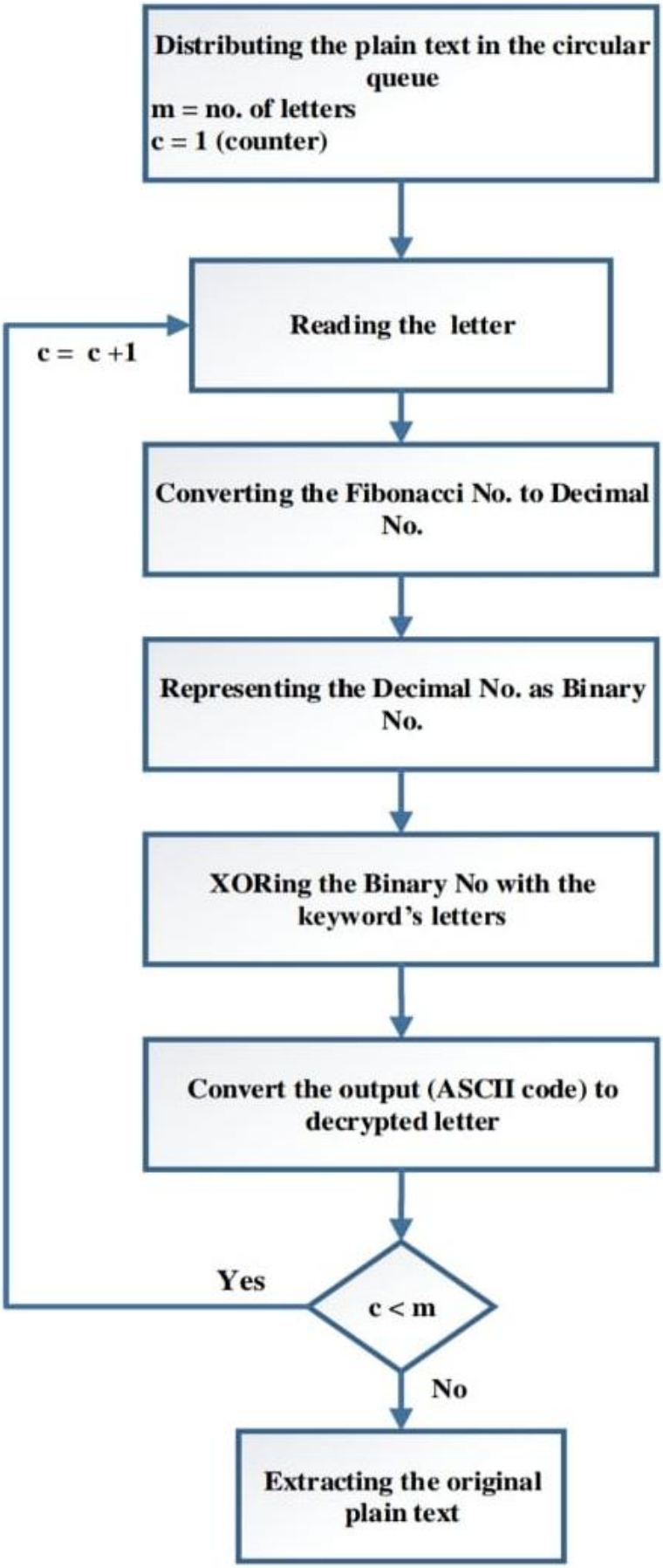
Example:

BINARY NUMBER	4	0	0	0	0	0	1	0	0
KEYWORD LETTER	E	0	1	0	0	0	1	0	1
DECRYPTION LETTER		0	1	0	0	0	0	0	1

Step 4: convert the output obtained from XORing which represents the ASCII code to the corresponding letter.

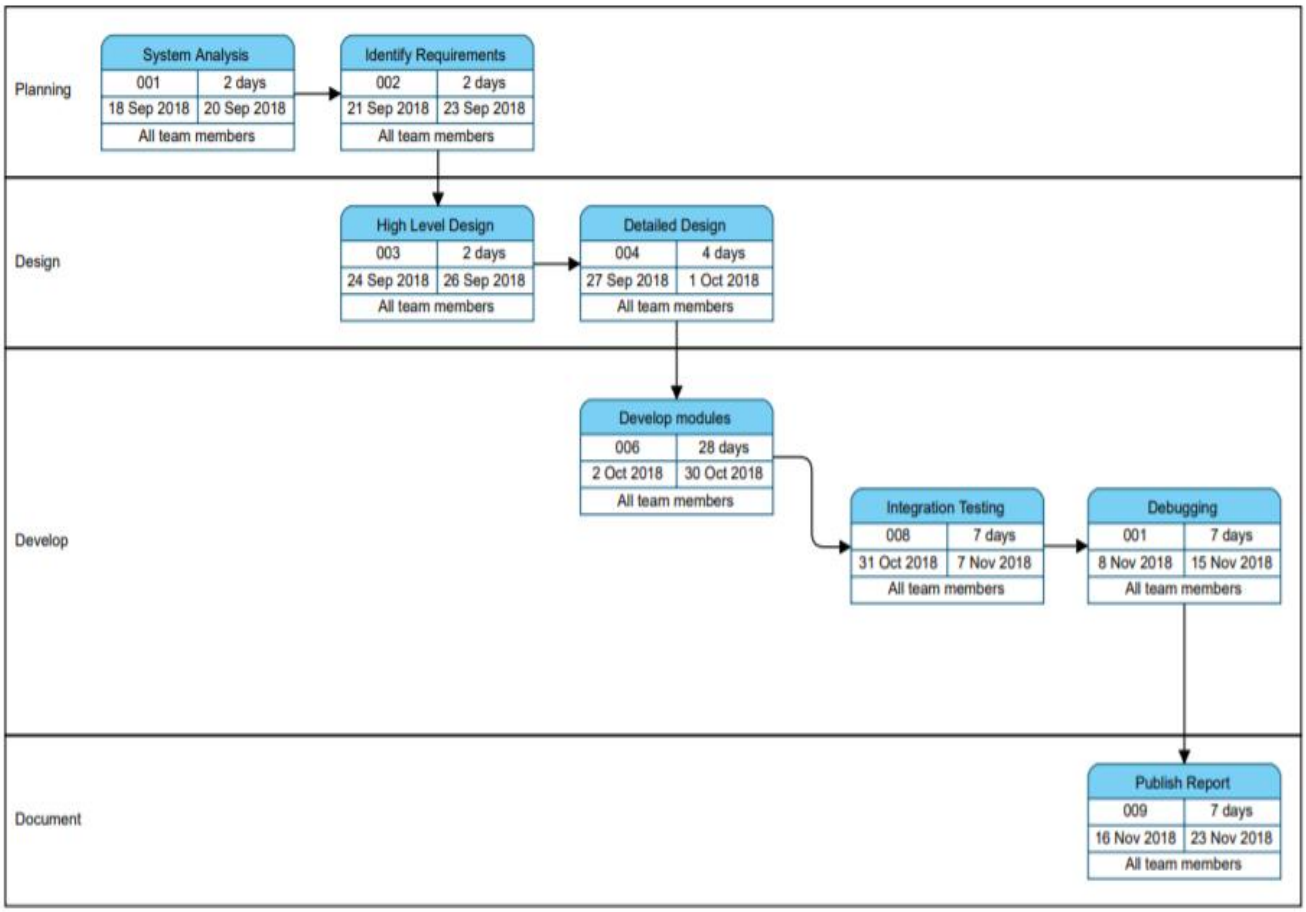
Example:

XORing OUTPUT (ASCII CODE)								LETTER
0	1	0	0	0	0	0	1	A



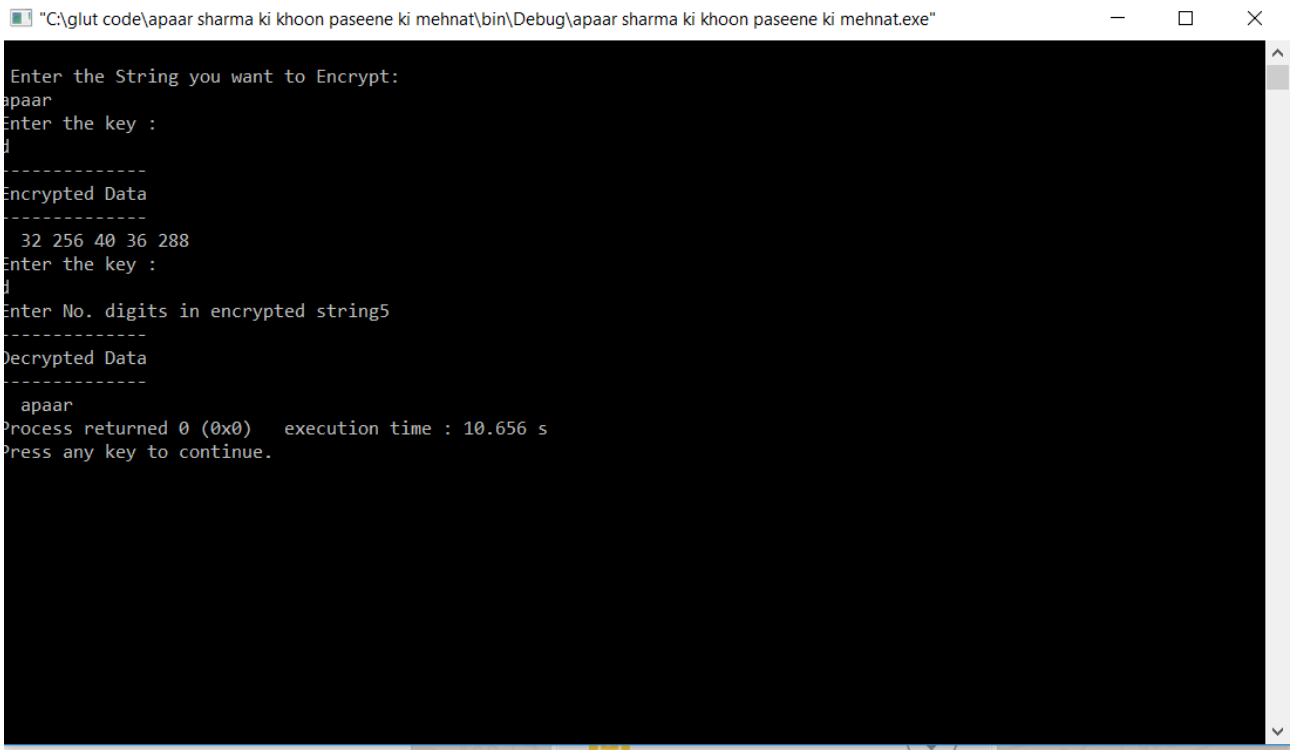
DECRYPTION PROCESS

7 PERT CHART

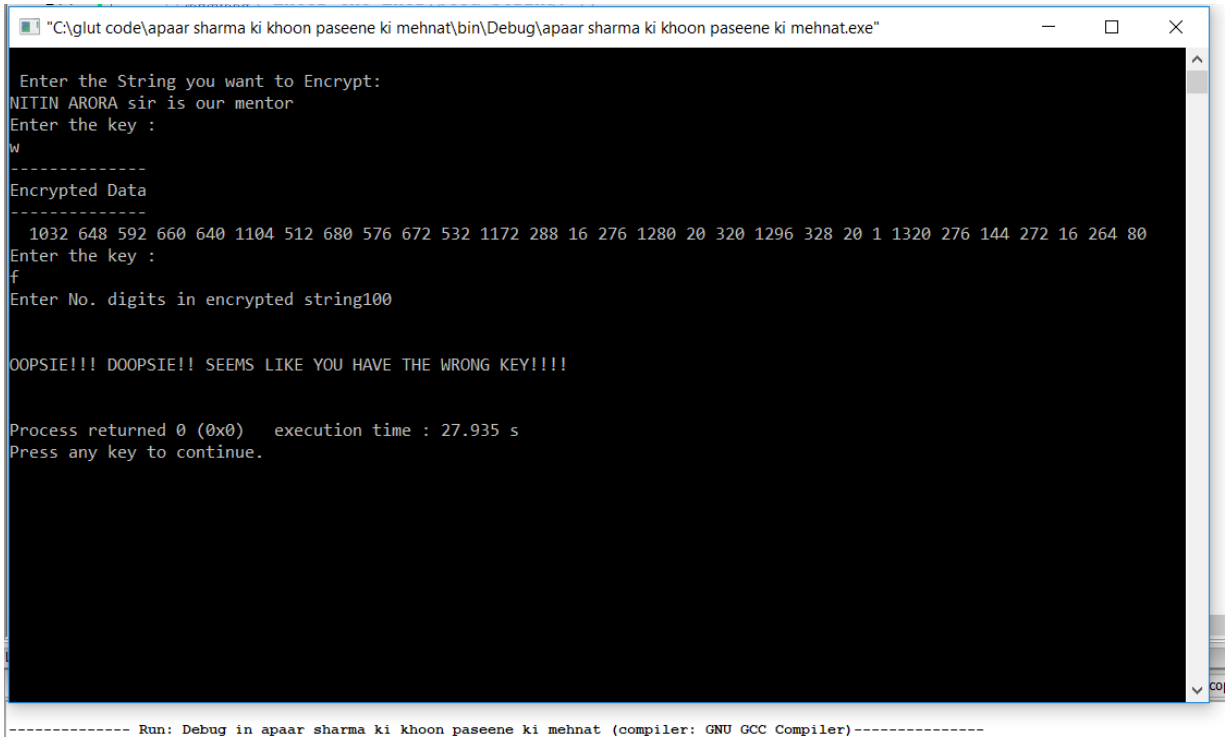

Planning
System Analysis
001
2 days
18 Sep 2018
20 Sep 2018
All team members
Identify Requirements
002
2 days
21 Sep 2018
23 Sep 2018
All team members
Design
High Level Design
003
2 days
24 Sep 2018
26 Sep 2018
All team members
Detailed Design
004
4 days
27 Sep 2018
1 Oct 2018
All team members
Develop
Develop modules
006
28 days
2 Oct 2018
30 Oct 2018
All team members
Integration Testing
008
7 days
31 Oct 2018
7 Nov 2018
All team members
Debugging
001
7 days
8 Nov 2018
15 Nov 2018
All team members
Document
Publish Report
009
7 days
16 Nov 2018
23 Nov 2018
All team members

OUTPUT:

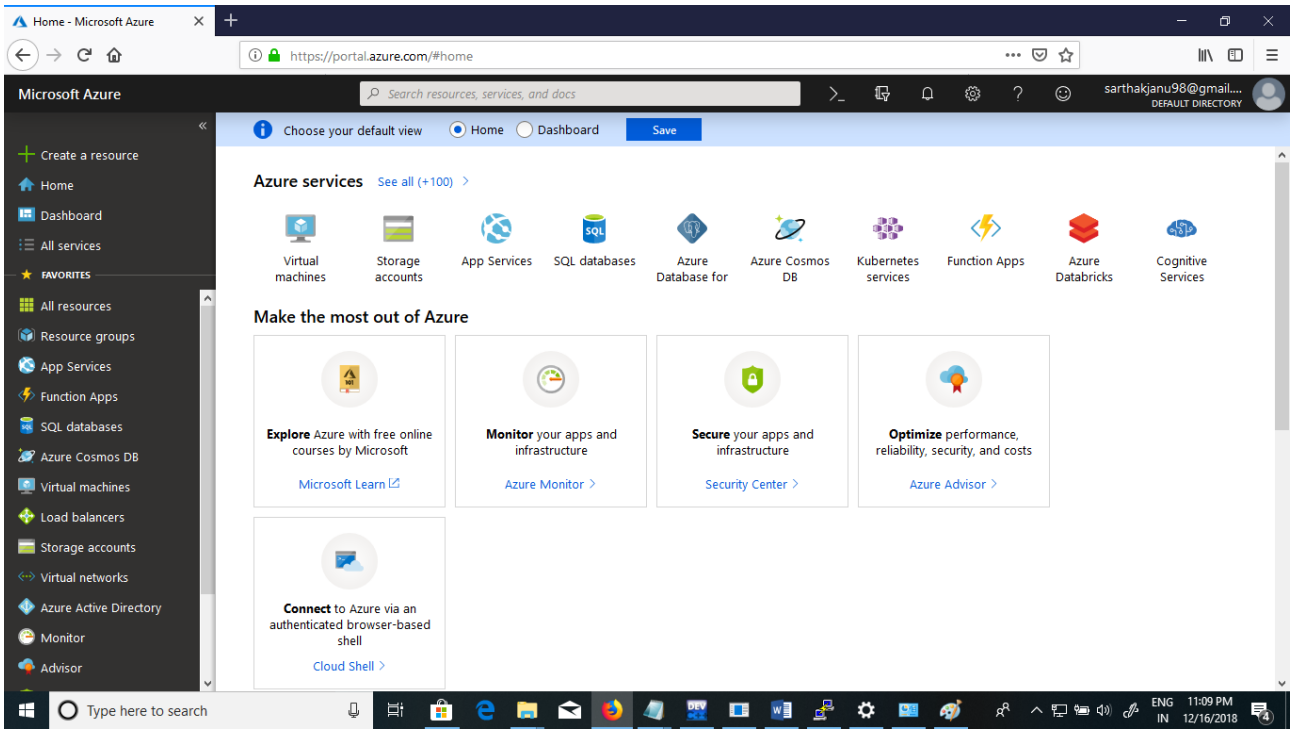
This output was taken on IDE codeblocks :



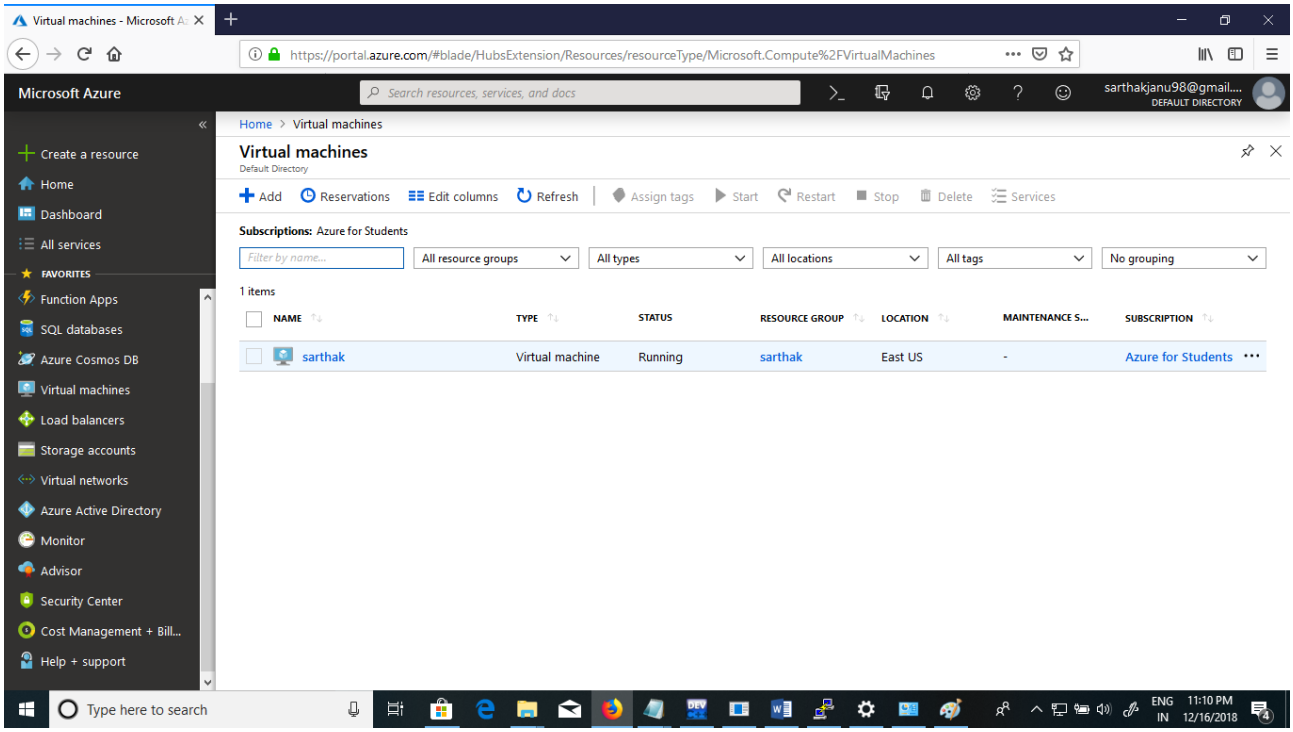
When you enter the wrong key



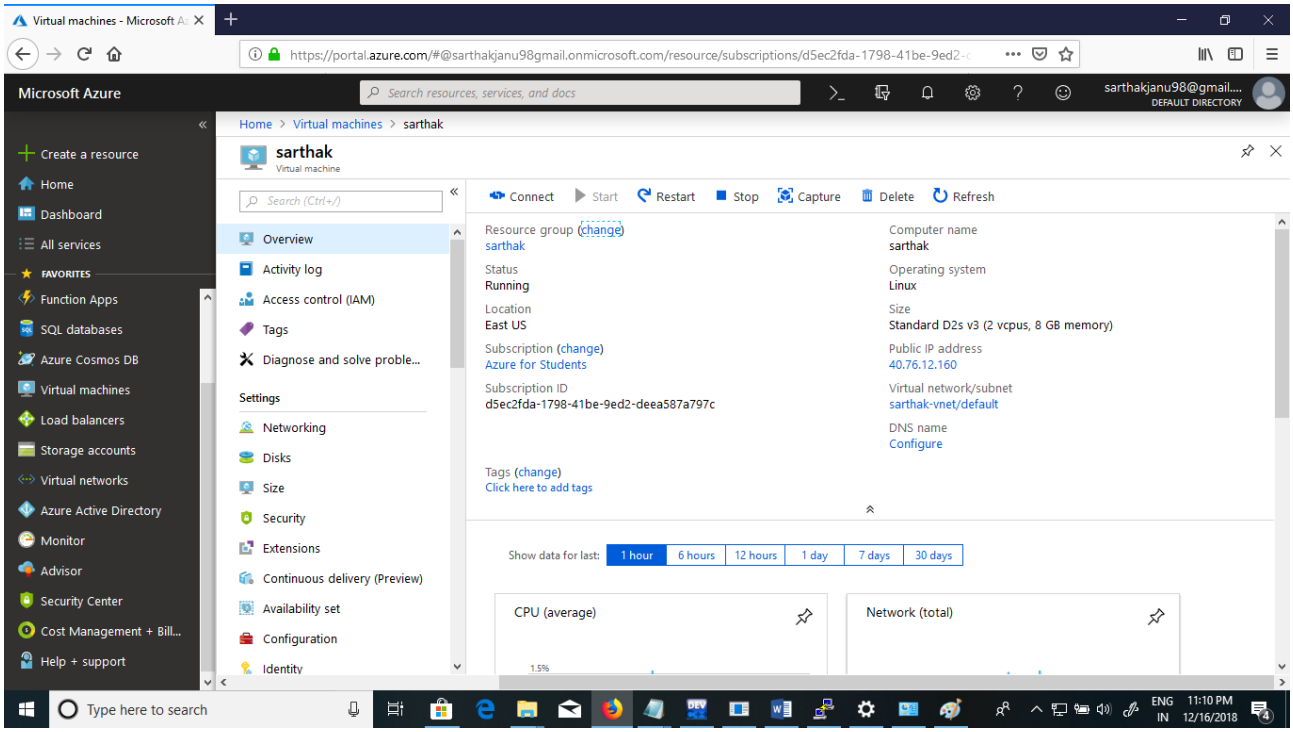
We set up an account on Microsoft azure



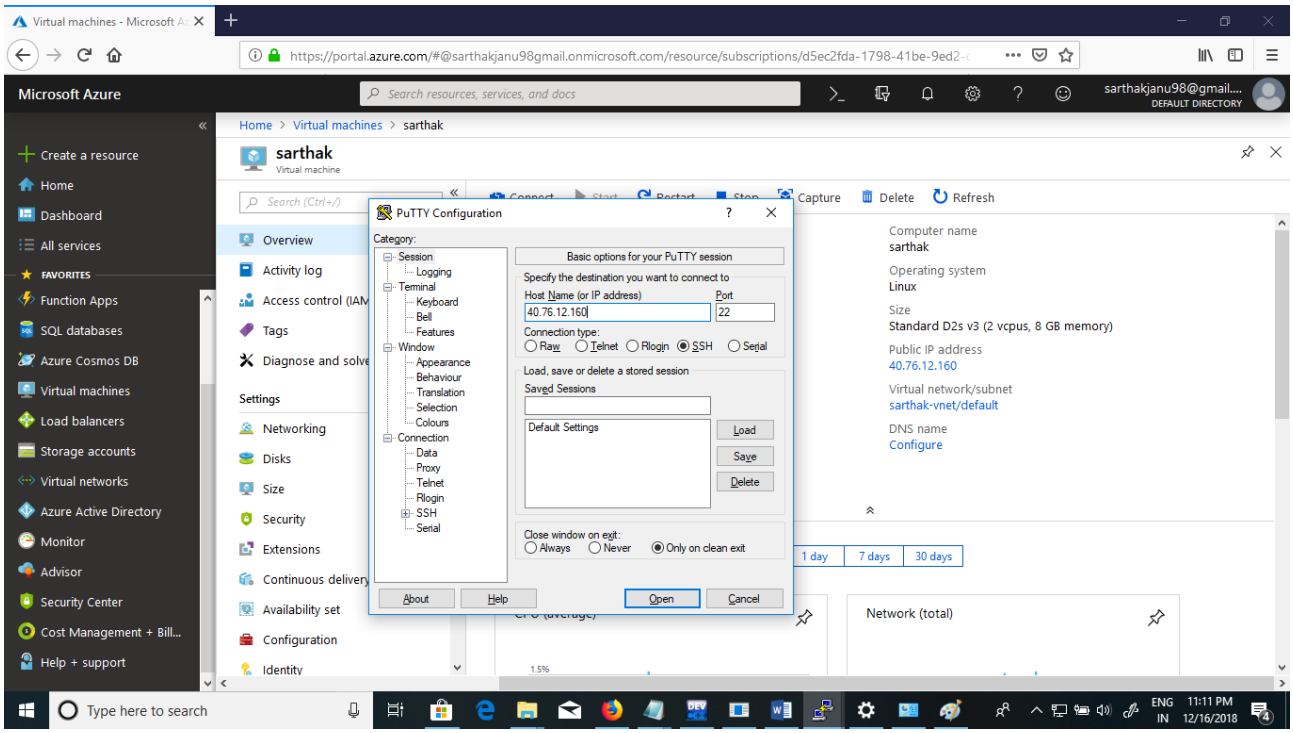
We set up a virtual machine on Microsoft azure



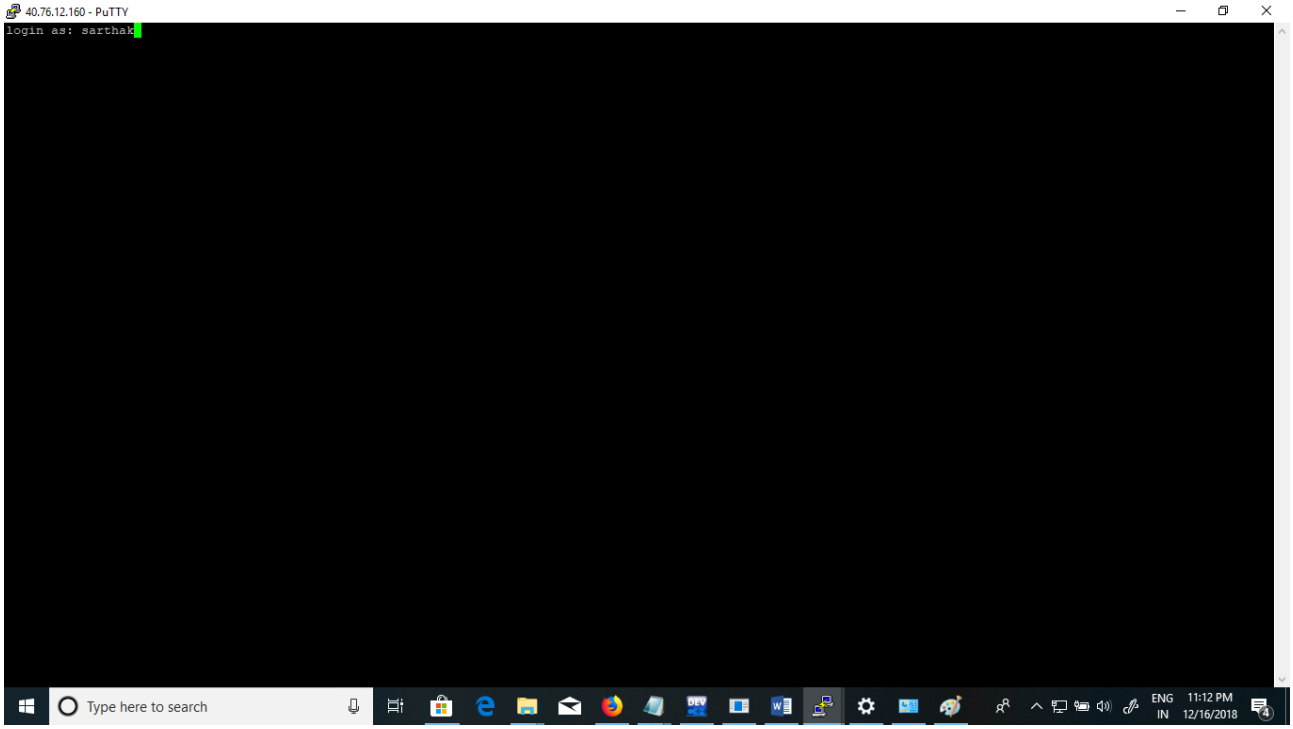
We set up the DNS and the unique IP for our hosted cloud



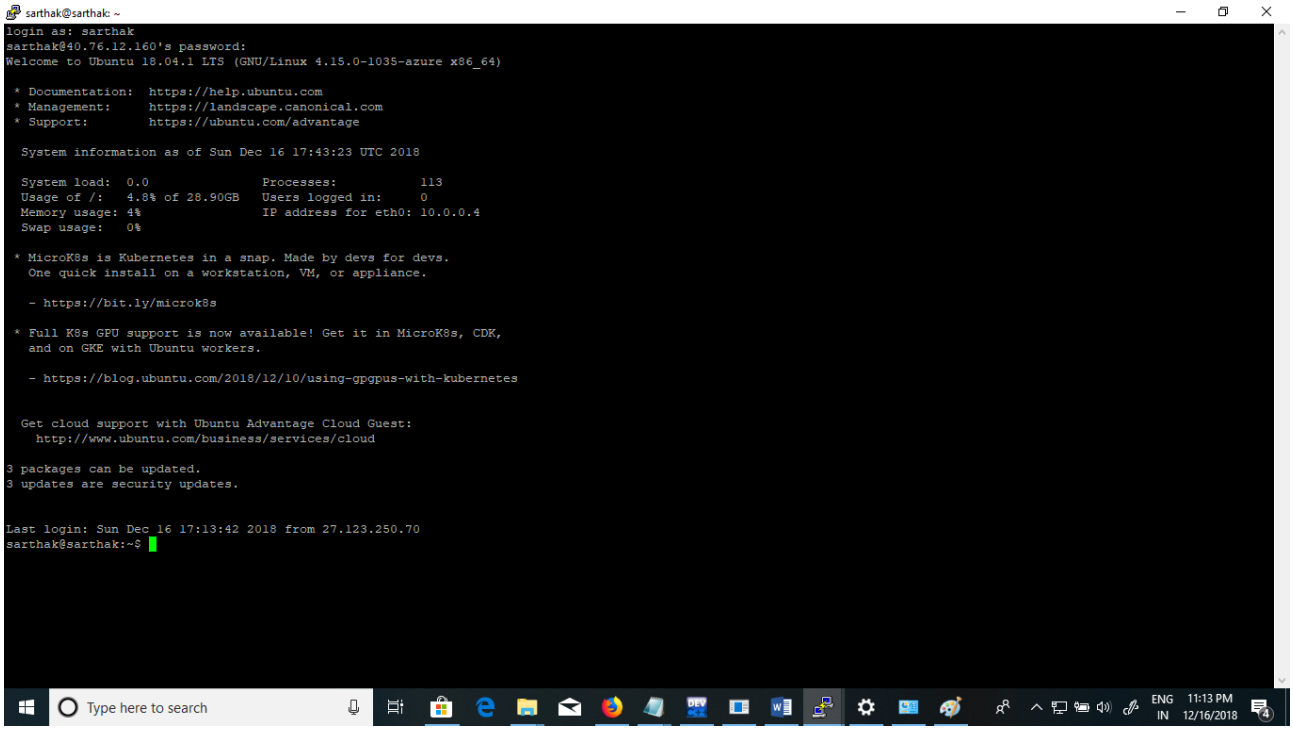
Now we entered the unique IP to puTTy



We entered our login id for the cloud on puTTy console



Login completed after entering the password



Now we are able to run our encryption/decryption code on the cloud

```
Last login: Sun Dec 16 17:13:42 2018 from 27.123.250.70
sarthak@sarthak:~$ ls
a.out  code.c  encrypt.c
sarthak@sarthak:~$ g++ encrypt.c
sarthak@sarthak:~$ ./a.out

 Enter the String you want to Encrypt:
qwertyuiopasdfghjhkjkk
Enter the key :
4
enecrypted code :577 552 660 577 640 545 648 657 660 641 1041 640 260 272 264 289 297 296 296 296 257 256 sarthak@sarthak:~$
```

Final output of the code running on the cloud

```
sarthak@sarthak:~$ g++ code.c
code.c: In function 'int main()':
code.c:20:30: warning: format '%[^
   ' expects argument of type 'char*', but argument 2 has type 'char (*)[100]' [-Wformat=]
         scanf("%[^\n]", &data);
                         ~~~~~^
sarthak@sarthak:~$ ./a.out
Enter 2 digits key: 12

Enter data to encrypt: NITIN ARORA

=============
Encrypted Data
=============
BEXEB,M^C^M

Enter unlock key: 12

=============
Decrypted Data
=============
NITIN ARORA

sarthak@sarthak:~$
```

10 References

[1] Wu, Suli, Yang Zhang, and Xu Jing. "A Novel Encryption Algorithm based on Shifting and Exchanging Rule of bi-column bi-row Circular Queue", IEEE International Conference on Computer Science and Software Engineering, Vol. 3. , 2008

[2] Amounas, Fatima., "An Elliptic Curve Cryptography based on Matrix Scrambling Method", IEEE International Conference on Network Security and Systems (JNS2), 2012.

[3] Merkle, Ralph C., and Martin E. Hellman. "On the Security of Multiple Encryption", Communications of the ACM, Vol. 24, No.7, 465-467, 1981.

[4] E. Barker, W. Barker, W. Burr, W. Polk and M. Smid, "Recommendation for Key Management-Part 1: General (Revision 3)", Computer Security Division (Information Technology Laboratory), 2016.

[5] Hankerson, Darrel, Alfred J. Menezes, and Scott Vanstone," Guide to Elliptic Curve Cryptography", Springer Science and Business Media, 2015.