

“Diffie-Hellman Key Exchange for Secure Text Transfer on Cloud”

PROJECT REPORT

Submitted for the course: Cyber Security (CSE4003)

By

NAME
NIVEDITA

REG. NO.
16BCE0451

Slot A2+TA2

Name of Faculty: PROF. Navamani T M
SCHOOL OF COMPUTER SCIENCE AND
ENGINEERING



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents

1. Abstract
2. Introduction
3. Literature Survey
4. Proposed System
 - 4.1 System Architecture
 - 4.2 Platform Details
 - 4.3 Modular Design
 - 4.3.1 MODULE 1 stand-alone-application
 - 4.3.2 MODULE 2 web-application
 - 4.4 Algorithm Details
 - 4.4.1 Diffie-Hellman
 - 4.4.2 AES Encryption
5. Implementation
 - 5.1 Planning and Analysis
 - 5.2 Screenshots
6. Conclusion
7. References

1.0 ABSTRACT

Cloud computing framework is accepted by most of the enterprises. It provides on-demand service, broad network access, elasticity etc. But the usability of these characteristic get obstructed by many security challenges. As the cloud service user uploads their confidential data over the cloud platform, it must be transmitted in a secure way.

Diffie-Hellman key exchange is a method of securely exchanging cryptographic keys over a public channel. It allows two parties that have no prior knowledge of each other to jointly establish a shared secret key. By using this method, you can double ensure that your secret message is sent secretly without outside interference of hackers or crackers (Man-In-The-Middle Attacks are possible though). If sender sends this cipher text in public others will not know what is it, and it will be received by receiver. The system uses online database to store all related information. As the project files and a database file will be stored into the Cloud, the project will be accessed in the web browser through the Cloud Services' link. This project enables the user to send text as secret messages and gives a key or a password to lock the text. The key encrypts the text, so that even if it is hacked by hacker, he will not be able to read the text. Receiver will need the key to decrypt the hidden text. User then sends the key to the receiver. Then the receiver presses decrypt key to get the secret text from the sender.

2.0 INTRODUCTION

Cloud security is one of the main concerns in the cloud computing domain. Storing personal and sensitive information on a third-party storage medium poses serious risks of data theft and data misuse by any person with malicious intent. As the cloud service user uploads their confidential data over the cloud platform, it must be transmitted in a secure way. In order to get rid of the same, a variety of encryption algorithms and mechanisms are used. Many researchers choose the best they found and use it in different combination to provide security to the data in cloud to ensure confidentiality, integrity and authentication while maintaining the efficiency and speed of the system. Diffie Hellman, along with a strong encryption algorithm can be used to make the users' Cloud data more secure. This project provides encryption solution between user applications and database servers in the cloud initiated by the user himself. The user can send text as secret messages and gives a key or a password to lock the text. The key encrypts the text, so that even if it is hacked by hacker, he will not be able to read the text. Receiver will need the key to decrypt the hidden text. User then sends the key to the receiver. Then the receiver presses decrypt key to get the secret text from the sender.

Diffie-Hellman Algorithm can be attacked by the Man-In-The-Middle attack. In this project, we aim to overcome this problem. This proposed architecture makes it tough for hackers to crack the security system, thereby protecting data stored in cloud. Our approach proposes a method that involves encrypting the file using any standard encryption technique and using Diffie Hellman for user authentication. In this way the files can be saved in a public domain securely without the threat of being used by any unauthorized person.

3.0 LITERATURE REVIEW

We referred to the following research papers related to our topic:

- 1.) Enhanced-Elliptic Curve Diffie Hellman Algorithm for Secure Data Storage in Multi Cloud Environment
- 2.) Efficient and Secure Data Sharing Using AES and Diffie Hellman Key Exchange Algorithm in cloud
- 3.) Security Algorithms for Cloud Computing
- 4.) Providing Data Security in Cloud Computing Environment using Diffie-Hellman and HMAC.

Paper 1

Title: Enhanced-Elliptic Curve Diffie Hellman Algorithm for Secure Data Storage in Multi Cloud Environment

Journal: International journal of Intelligent Engineering and systems.

Authors: Anitha Patil

Year: January 16,2018

Algorithms used:

1. Diffie-Hellman.

Drawbacks:

1. Concerns only with data storage.
2. **Encryption/decryption algorithms not used.**

Paper 2

Title: Efficient and Secure Data Sharing Using AES and Diffie Hellman Key Exchange Algorithm in cloud

Journal: Middle-East Journal of Scientific Research

Authors: Grace Sophia and Prabakeran

Year: 2016

Algorithms used:

AES and Diffie Hellman Key Exchange Algorithm

Drawbacks:

1. A limited amount of cipher text classes can be used.
2. In cloud, the count of cipher texts are increased rapidly.
3. So maximum cipher classes have to be allocated. Otherwise, public-key value can be increased.

Paper 3

Title: Security Algorithms for Cloud Computing

Journal: International Conference on Computational Modeling and Security

Authors: Akashdeep Bhardwaja, Vinay Avasthic , Hanumat Sastryd

Year: 2016

Alogorithms used:

AES, RSA, Triple DES and MD5

Drawbacks:

1. RSA has very slow key generation, signing and decryption which is slightly tricky to implement.
2. AES is also complex to implement in softwares taking both performance and security into consideration.

Paper 4

Title: Providing Data Security in Cloud Computing Environment using Diffie-Hellman and HMAC.

Journal: International Journal of Computer Science and Computing

Authors: Pooja, Dr.Dheerendra Singh

Year: 2017

Alogorithms used:

Fully Homomorphism Encryption, HMAC, Diffie-Hellman

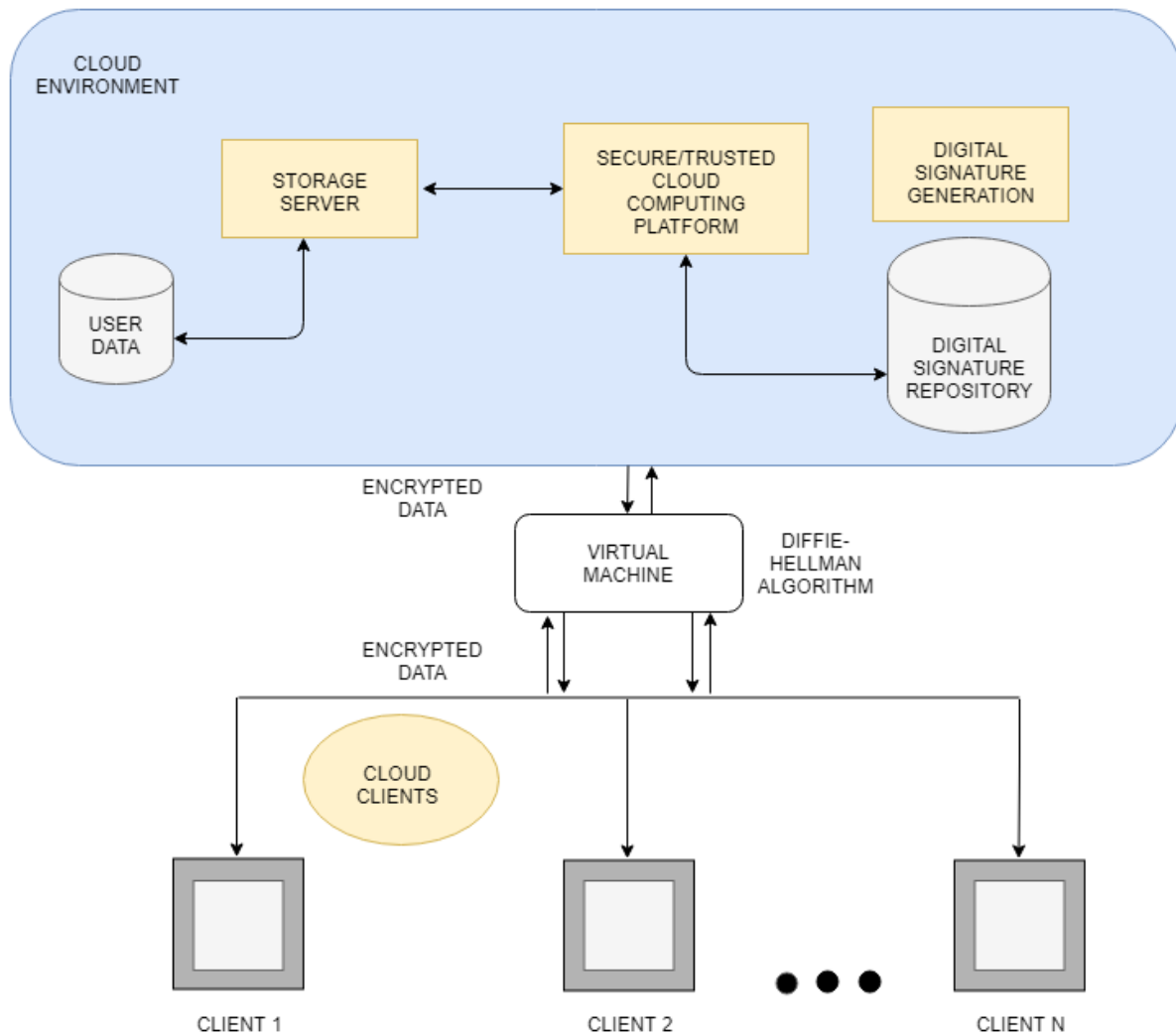
Drawbacks:

1. The same encryption scheme may be applied to the SQL queries for cloud database
2. Or the load of virtual machine can be reduced by adding any other factor without compromising the security of the data

4.0 PROPOSED SYSTEM

This project provides encryption solution between user applications and database servers in the cloud initiated by the user himself. The user can send text as secret messages and gives a key or a password to lock the text. The key encrypts the text, so that even if it is hacked by hacker, he will not be able to read the text. Receiver will need the key to decrypt the hidden text. User then sends the key to the receiver. Then the receiver presses decrypt key to get the secret text from the sender. Diffie-Hellman Algorithm can be attacked by the Man-In-The-Middle attack. In this project, we aim to overcome this problem. This proposed architecture makes it tough for hackers to crack the security system, thereby protecting data stored in cloud. Our approach proposes a method that involves encrypting the file using any standard encryption technique and using Diffie Hellman for user authentication. In this way the files can be saved in a public domain securely without the threat of being used by any unauthorized person.

4.1 ARCHITECTURE DIAGRAM



4.2 PLATFORM DETAILS

This project is developed using:

- AWS
- Cloud Database
- Python 3.0
- Python libraries like:
 - Tkinter
 - Crypto
 - Secretsharing
 - Flask

4.3 MODULAR DESIGN

This implementation can be explained in two modules

MODULE 1 stand-alone-application

MODULE 2 web-application

MODULE 1

stand-alone-application

- This portion deals with encryption and decryption of file
- The file is encrypted using AES algorithm
- **src/stand-alone-application/DH.py**: This file deals with generating keys using diffie-hellman. It generates three keys: Private Key, Public Key, Secret Key (used for encryption and decryption)
- **src/stand-alone-application/ENCDEC.py**: This file is used for encoding and decoding using AES algorithm.
- **src/stand-alone-application/thrain.py**: This file acts as a mediator and connect the main program with other code files.
- **src/stand-alone-application/main.py**: This file deals with the GUI. It is the main file [yeah, trust me!].

MODULE 2

web-application

Once file is encrypted it has to be uploaded on an online directory. Another directory is needed where public-key of all the users is stored. Thus, we built an online directory and hosted it on cloud. The unique thing about hosting is that dynamic files are being generated while adding a new user or uploading a text file. Thus, we needed a cloud service which could run the program and incorporate the dynamic files. Where amazon AWS is used.

Amazon Web Services

Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. The technology allows subscribers to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, harddisk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, CRM, etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser.

Elastic Compute Cloud [EC2]

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios. For

compute, AWS' main offering is its EC2 instances, which can be tailored with a large number of options. It also provides related services such as Elastic Beanstalk for app deployment, the EC2 Container service, AWS Lambda and Autoscaling. In general terms, prices are roughly comparable, especially since AWS shifted from by-the-hour to

by-the-second pricing for its EC2 and EBS services in 2017. This economic pricing, reliable and secure infrastructure and vast online support enabled us to use Amazon to host and deploy the service on it is IaaS platform.

Python

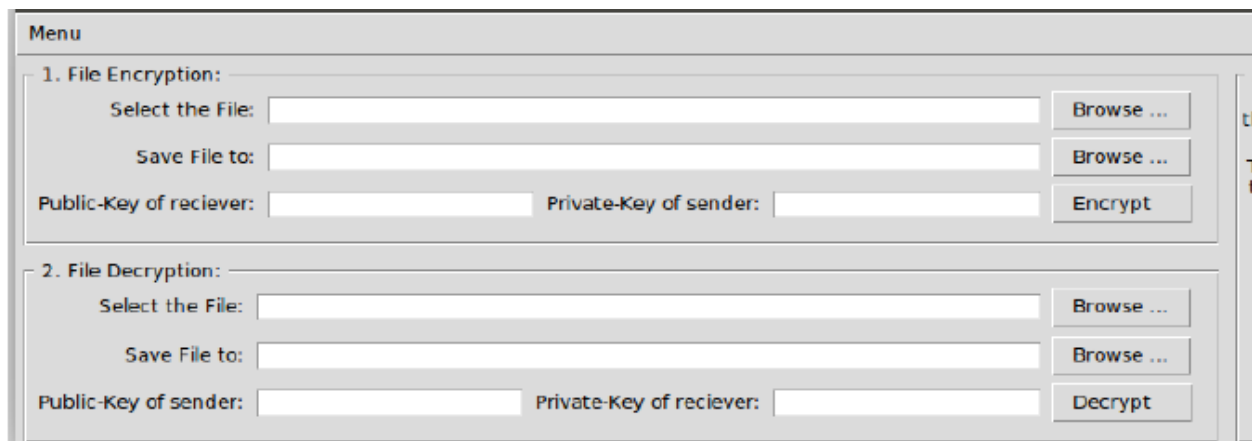
Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespaces. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative and procedural, and has a large and comprehensive standard library.

Such large and comprehensive standard libraries along with the documented support helped us to choose python as the language which we used to build both, the stand alone as well as web-based application.

Below is a brief description of the frameworks and libraries extensively used to build the desired framework.

Python Tkinter

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin objectoriented layer on top of Tcl/Tk. Tkinter is not the only GUI Programming toolkit for Python. It is however the most commonly used one. Cameron Laird calls the yearly decision to keep Tkinter "one of the minor traditions of the Python world." Tkinter is a GUI (graphical user interface) widget set for Python. This document was written for Python 2.7 and Tkinter 8.5 running in the X Window system under Linux. Tkinter helps users to build a cross-platform application and is easy to use, thus, we used it to build the GUI of our stand-alone application.



GUI built using python-tkinter

Python crypto

Another python's extensively maintained library is PyCrypto. It is an actively developed library that provides cryptographic recipes and primitives. It provides secure hash functions and various encryption algorithms. Hash functions played an important role in the formation of the

framework. The main function of the hash function was to generate a private key for new users, which was less than the pre-defined prime number. Library hashlib was deployed for the cause, where a variable length was being passed to the function and a digest was generated. This library was also useful as it already had implementation of various symmetric- and asymmetric-key encryption algorithms like AES, blowfish, ARC2 and many more. Encryption algorithms transform plaintext in some way that is dependent on a key or key pair, producing ciphertext. We employed AES and blowfish (symmetric-key encryption) for encrypting the text. We performed a double layered encryption.

Python flask

Flask is a BSD licensed microframework for Python based on Werkzeug and Jinja 2. “Micro” does not mean that your whole web application must fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The “micro” in microframework means Flask aims to keep the core simple but extensible. Flask won’t make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to you, so that Flask can be everything you need and nothing you don’t.

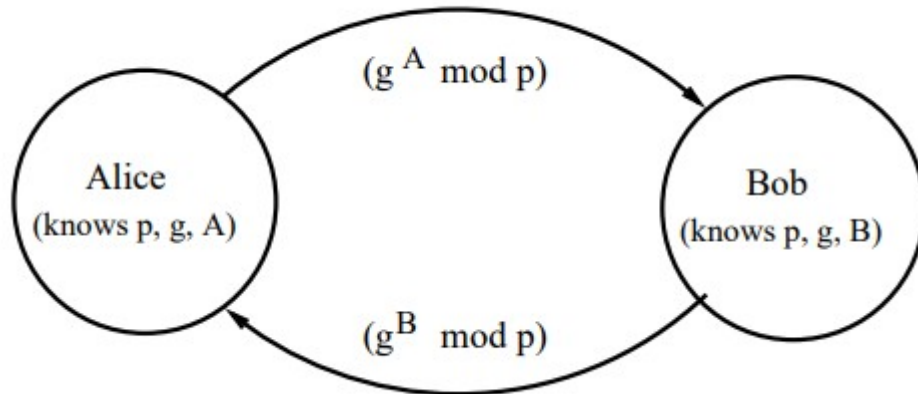
4.4 ALGORITHM DETAILS

4.4.1 Diffie-Hellman

Diffie Hellman key exchange offers the best of both as it uses public key techniques to allow the exchange of a private encryption key. By using this method, you can double ensure that your secret message is sent secretly without outside interference of hackers or crackers (Man-In-The-Middle Attacks are possible though).

This project aims to make the data stored in Cloud secure by using Diffie-Hellman algorithm

The point is to agree on a key that two parties can use for a symmetric encryption, in such a way that an eavesdropper cannot obtain the key.



Steps in the algorithm:

Working example of Diffie-Hellman

p is a prime number

g is a primitive root modulo of p

- Alice and Bob agree to use a modulus $p = 23$ and base $g = 5$
- Alice gets her private key (key which she should not share with anyone) generated as 4.
- Thus, public key generated for Alice shall be $5^4 \% 23 = 625 \% 23 = 4$
- Bob gets his private key (key which he should not share with anyone) generated as 3.
- Thus, public key generated for Bob shall be $5^3 \% 23 = 125 \% 23 = 10$
- Now, Alice gets the public key of Bob and generates a secret key. i.e. $(\text{public key of Bob} \wedge \text{Private Key of Alice}) \bmod p \Rightarrow (10^4) \% 23 \Rightarrow 10000 \% 23 \Rightarrow 18$
- On the other side, Bob also uses a similar method to generate a secret key i.e. $(\text{public key of Alice} \wedge \text{Private Key of Bob}) \bmod p \Rightarrow (4^3) \% 23 \Rightarrow 64 \% 23 \Rightarrow 18$

Clearly, much larger values of a , b , and p are required. An eavesdropper cannot discover this value even if she knows p and g and can obtain each of the messages.

SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available. SHA-256 is not much more complex to code than SHA-1, and has not yet been compromised in any way. The 256-bit key makes it a good partner-function for AES. That's why we have used it to create a strong key for encryption.

4.4.2 Encryption

Encryption is widely used on the internet to protect user information being sent between a browser and a server, including passwords, payment information and other personal information that should be considered private. Organizations and individuals also commonly use encryption

to protect sensitive data stored on computers, servers and mobile devices like phones or tablets. There are various encryption techniques that are present some of which are:

- Triple DES
- Blowfish
- RSA
- Twofish
- AES

The technique that we have used in our project is AES and it is described below.

Advanced Encryption Standard

The more popular and widely adopted symmetric encryption algorithm nowadays is the Advanced Encryption Standard (AES). It is found to be at least six time faster than triple DES. A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback, but it was found to be slow.

The AES has three fixed 128-bit block ciphers with cryptographic key sizes of 128, 192 and 256 bits. Key size is unlimited, whereas the block size maximum is 256 bits. The AES design is based on a substitution-permutation network (SPN) and does not use the Data Encryption Standard (DES) Feistel network. The diagram below shows the implementation of AES encryption technique.

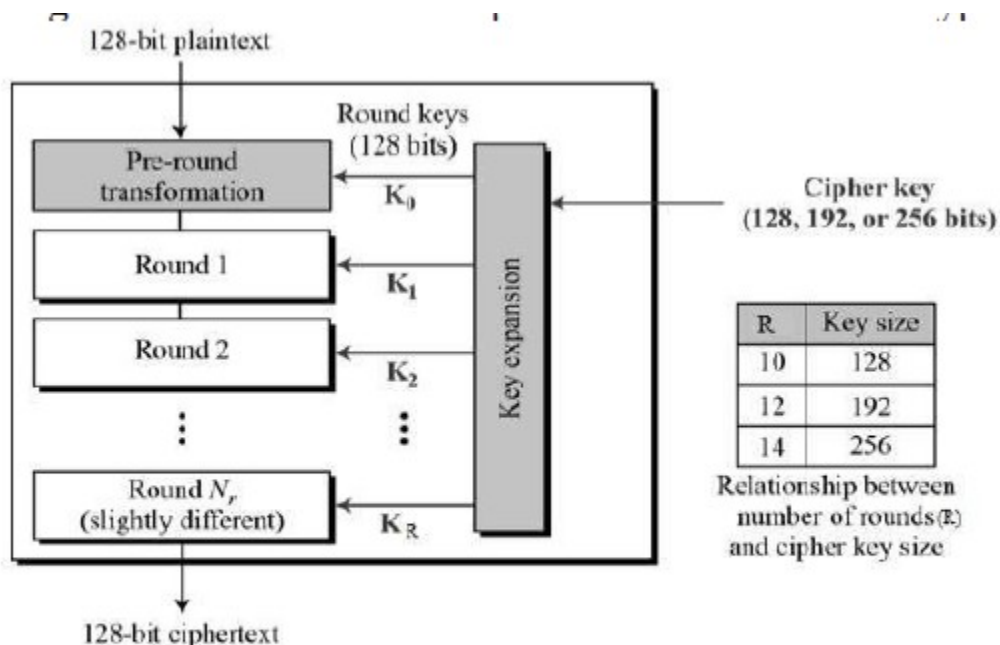


Figure 2: The schematic of AES structure

5.0 IMPLEMENTATION

5.1 Planning and Analysis

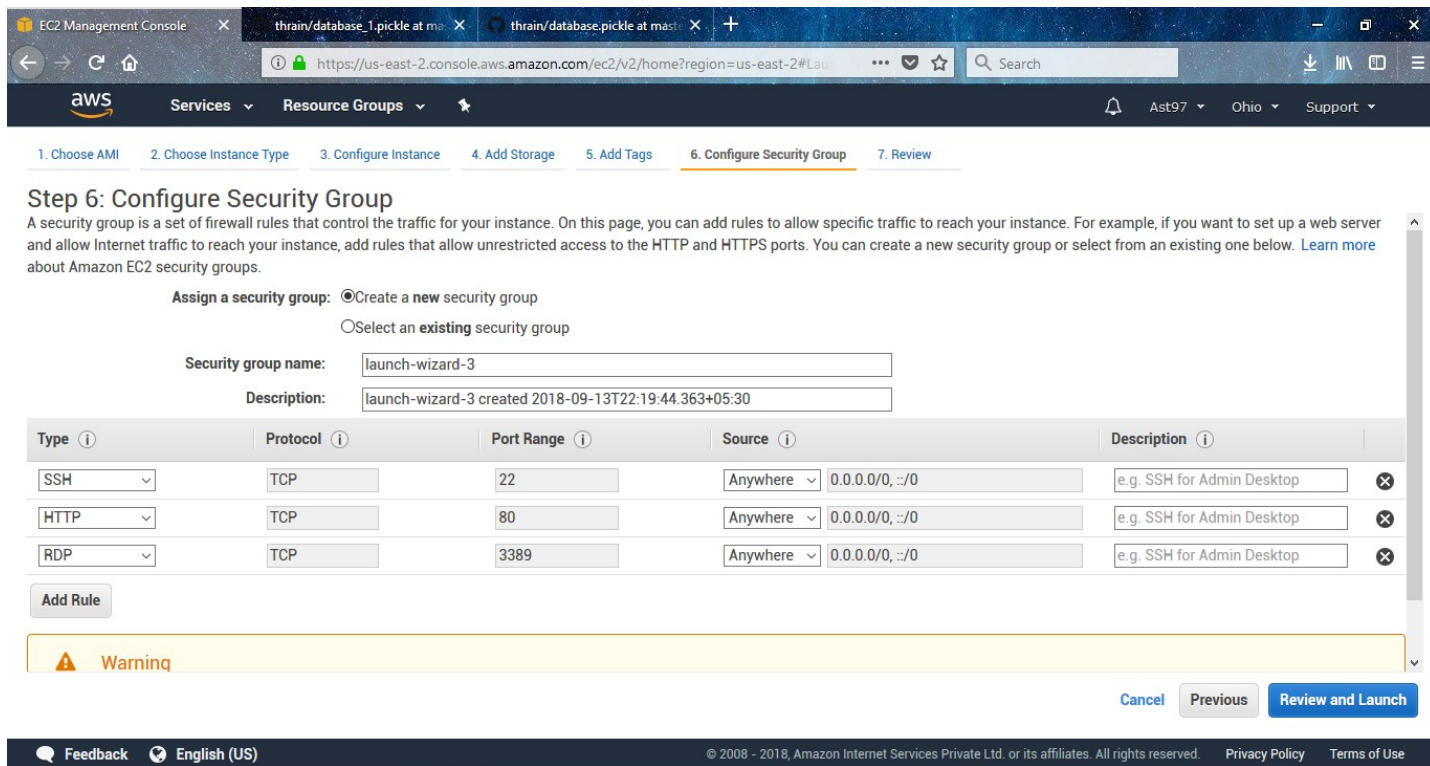
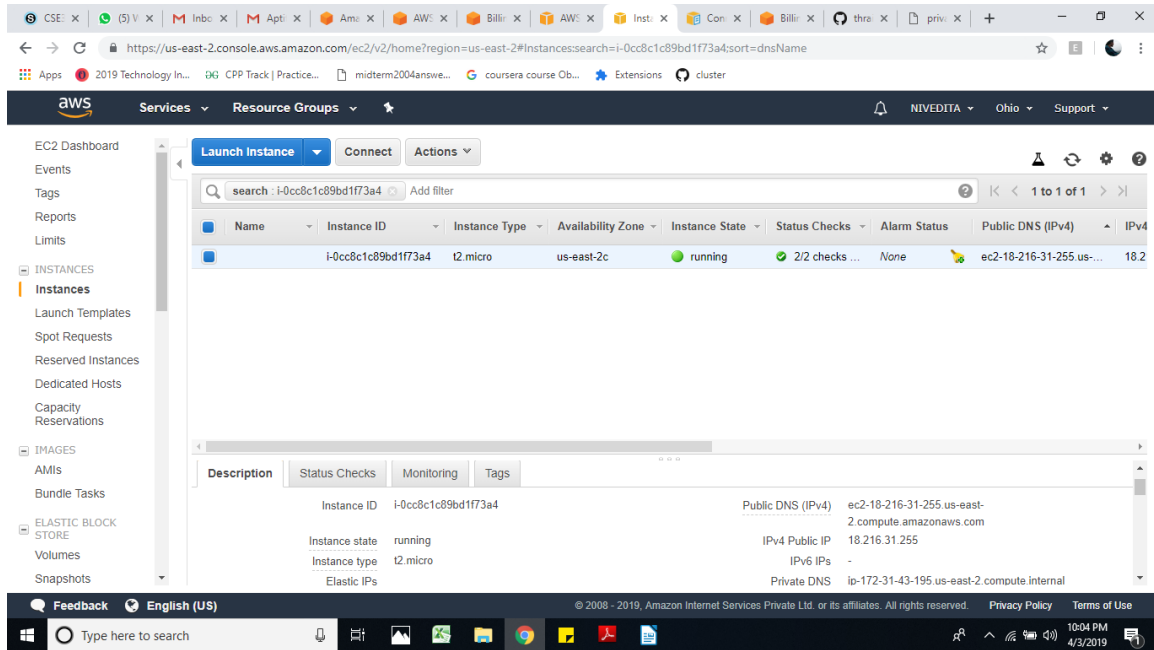
The main task of this project was to provide as secure a file storage on the cloud as possible. So, several issues had to be sorted out like :

- ➔ Where the file needs to be encrypted.?
- ➔ How the user must be authenticated?
- ➔ What will be the key for AES encryption?
- ➔ How this key will be related to the Diffie Hellman Key exchange protocol?

I first thought of encrypting the text online, but the *man in the middle attack* made me not choose that method. I also learnt that about an attack abbreviated as NFS. NFS in an attack in which a key of order 2^{768} could be computed. This was approximately 232 digits. Thus, came to a conclusion that a larger prime number is needed in the process. Therefore, used a prime number having 600 digits. While analysing all these questions we came upon this course of action. To provide greater control to the owner of the file, the file will be implemented on the owner's computer itself using an application. The Diffie Hellman was used to generate the file and only those users who have the final same key from this process would be able to decrypt the file. The final key generated from Diffie Hellman, being same for both intended participants, was used as the basis for the key for AES encryption.

The following sub-sections describe the different parts of the project in greater detail.
Cloud Connectivity

- **Connecting to AWS and making an Instance**



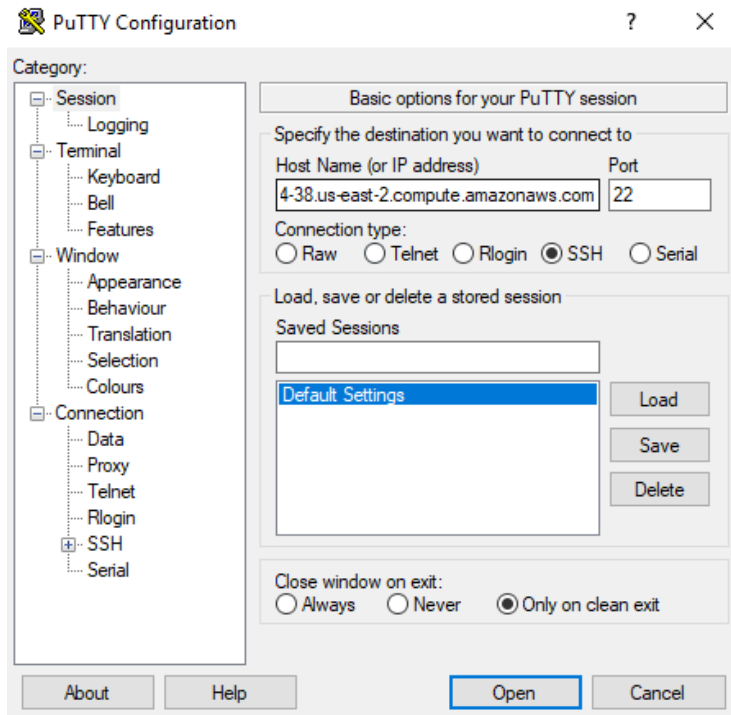
Instance ID: i-0cc8c1c89bd1f73a4

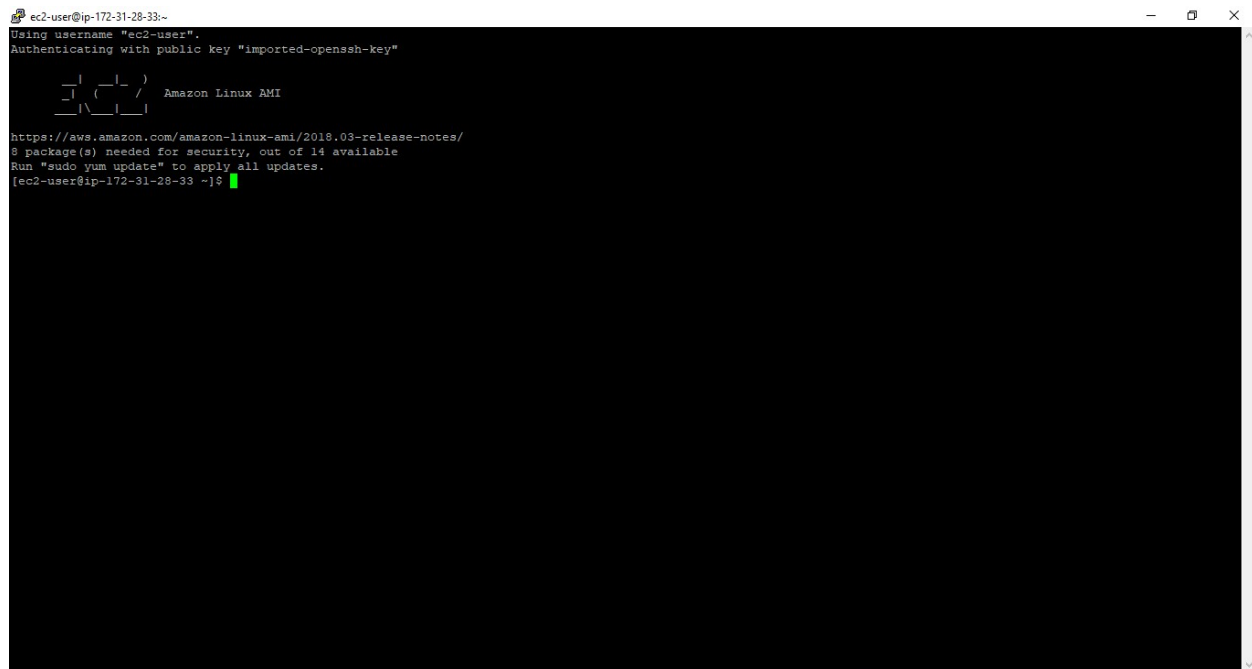
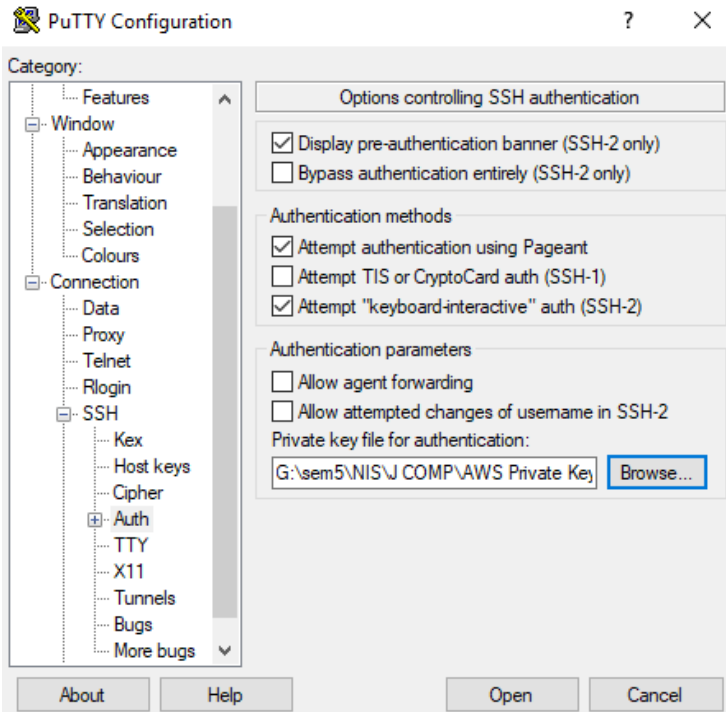
Public DNS IPV4: ec2-18-216-31-255.us-east-2.compute.amazonaws.com

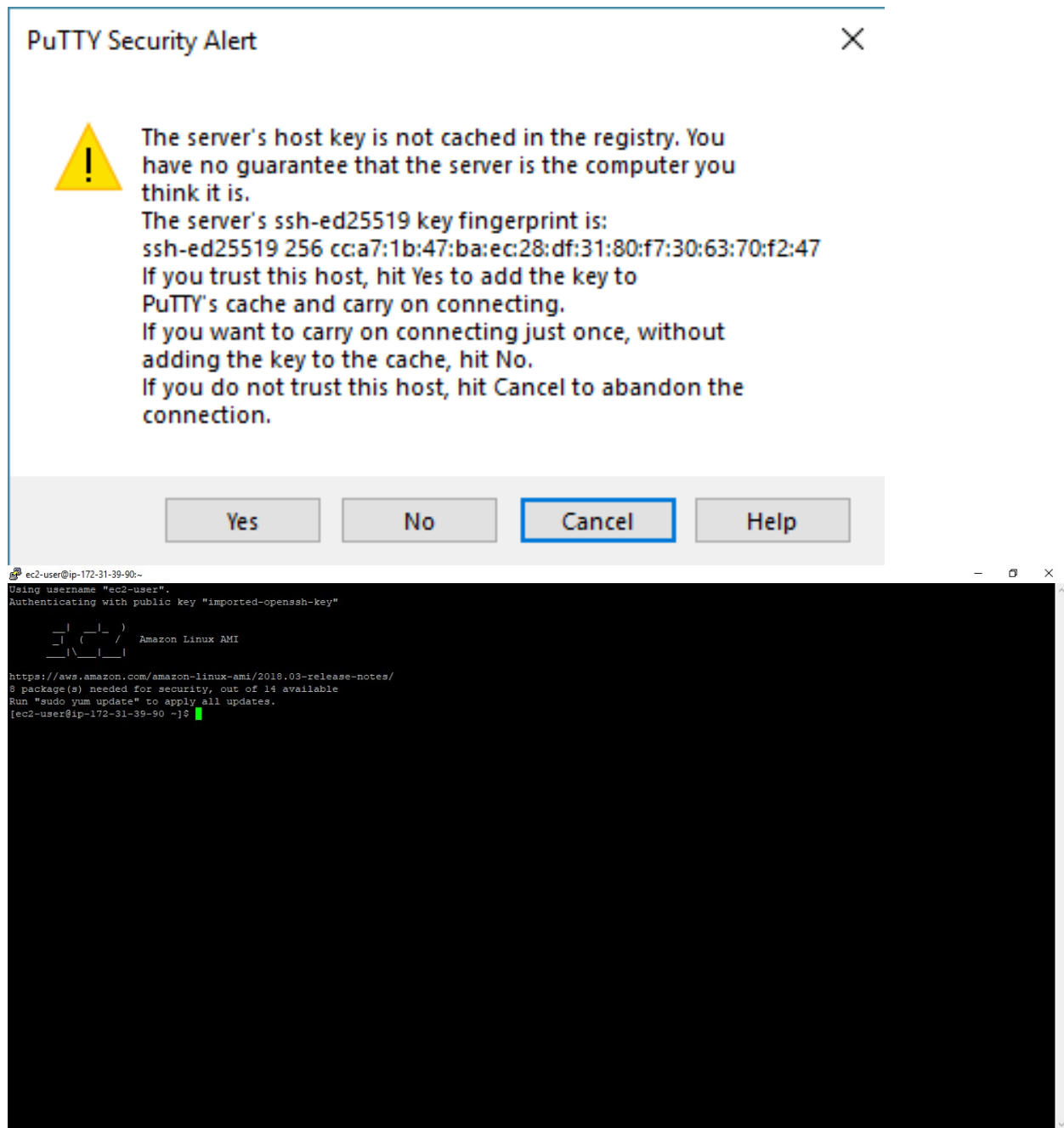
IPv4 Public IP: 18.216.31.255

To connect: ec2-user@ec2-18-216-31-255.us-east-2.compute.amazonaws.com

Editing the Putty Configuration for further use and adding ec2-user hostname.







The workings of the system:

Getting the code on the cloud:

Running the code after changing directory:

On opening the link, we get the following web page:

```
PuTTY (inactive)
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 68 (delta 8), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (68/68), done.
[ec2-user@ip-172-31-43-195 ~]$ cd thrain-master
-bash: cd: thrain-master: No such file or directory
[ec2-user@ip-172-31-43-195 ~]$ ls
cyber-security  thrain
[ec2-user@ip-172-31-43-195 ~]$ cd thrain
[ec2-user@ip-172-31-43-195 thrain]$ ls
dump  readme.md  Report.pdf  src  theory
[ec2-user@ip-172-31-43-195 thrain]$ cd src/web-application
[ec2-user@ip-172-31-43-195 web-application]$ python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
[ec2-user@ip-172-31-43-195 web-application]$ vi thrain/src/web-application/app.py
[ec2-user@ip-172-31-43-195 web-application]$ vi app.py
[ec2-user@ip-172-31-43-195 web-application]$ python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 190-253-566
[ec2-user@ip-172-31-43-195 web-application]$ python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 190-253-566
```

Options for the user x +

127.0.0.1:5000

Apps 2019 Technology In... CPP Track | Practice... midterm2004answe... coursera course Ob... Extensions cluster

Secure Message Transfer OPTIONS UPLOAD FILES FILE DIRECTORY DOWNLOAD PUBLIC KEY REGISTER USER

Secure Message Transfers Using Diffie-Hellman Key Exchange

The user gets to choose from one of the following options:

- He gets to upload the file to the cloud in a secured manner.
- He gets to choose from the list of all the different files that are present on the cloud provided he has the required credentials for decrypting the file.
- He can also download public key associated with the user with whom the file transfer is to take place.
- To upload his file to cloud storage the user needs to be registered so that his file can be shared with the other desired person in an instant.

Type here to search

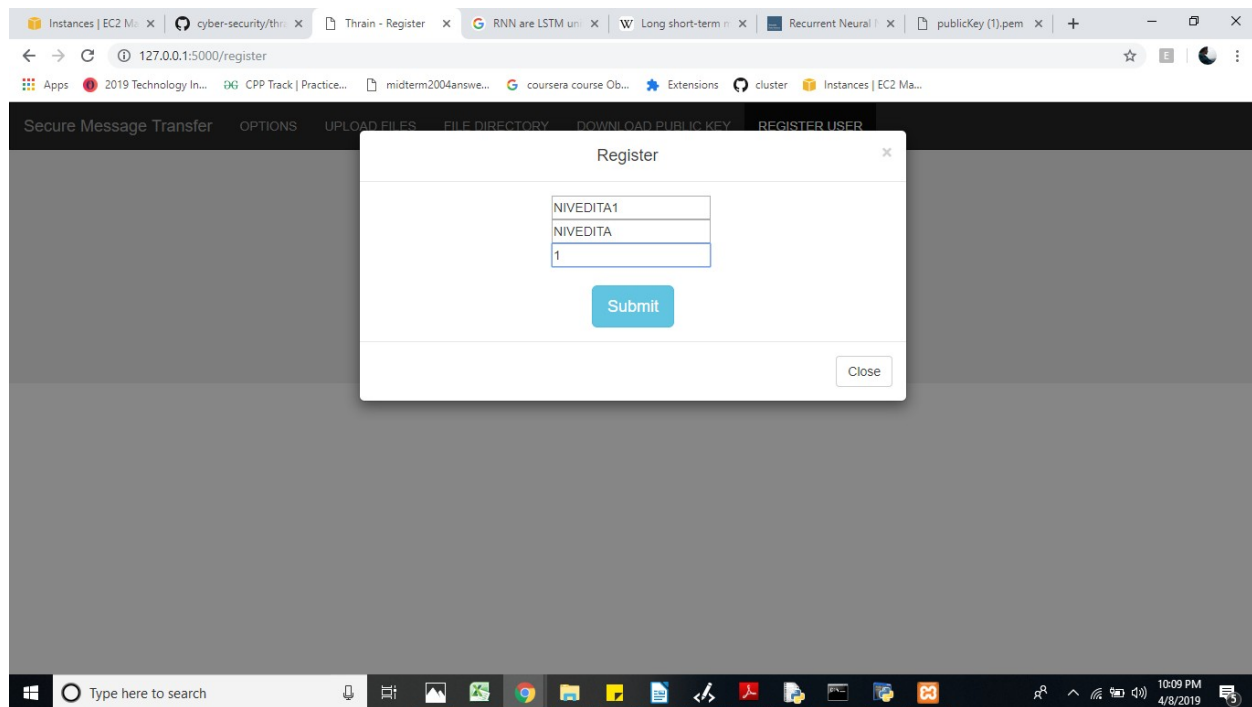
11:15 PM 4/7/2019

127.0.0.1:5000/register

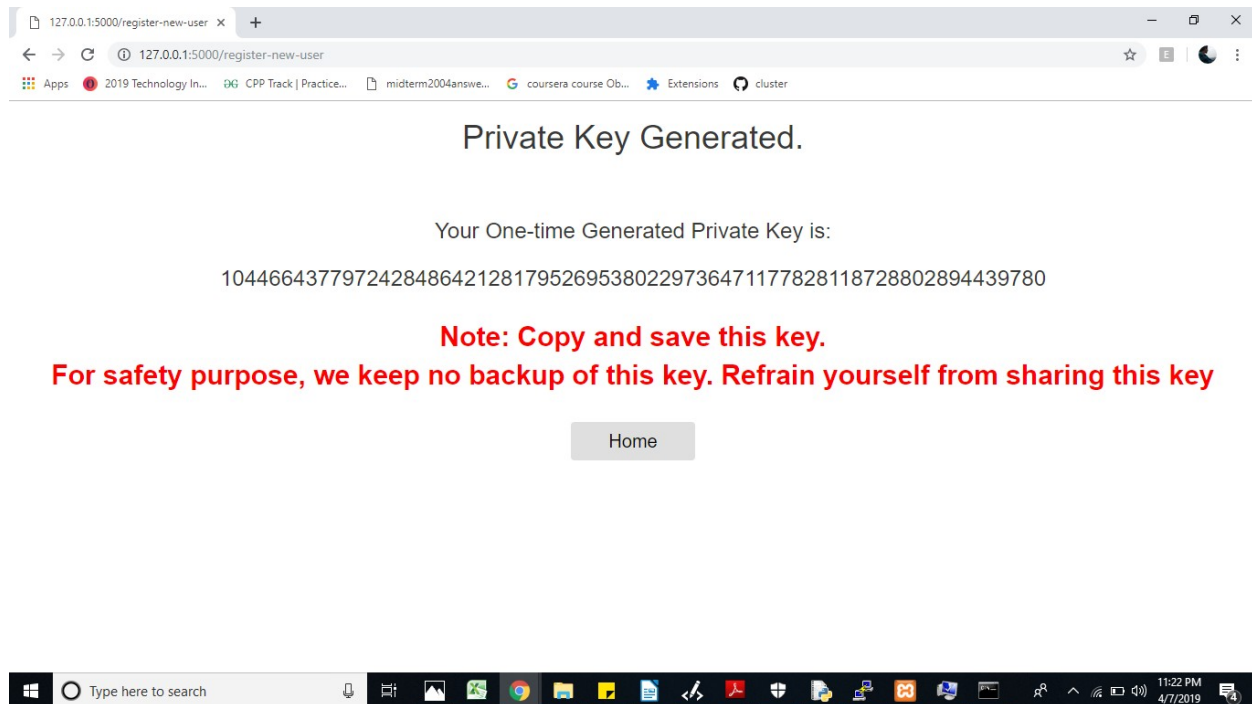
Secure Message Transfer OPTIONS UPLOAD FILES FILE DIRECTORY DOWNLOAD PUBLIC KEY REGISTER USER

REGISTER YOURSELF

Click Here to Register Yourself



Private key generated for the user “Nivedita”



Download Public Key:

Public Key List

127.0.0.1:5000/public-key-directory/

Apps 2019 Technology In... 0G CPP Track | Practice... midterm2004answe... coursera course Ob... Extensions cluster

Secure Message Transfer OPTIONS UPLOAD FILES FILE DIRECTORY **DOWNLOAD PUBLIC KEY** REGISTER USER

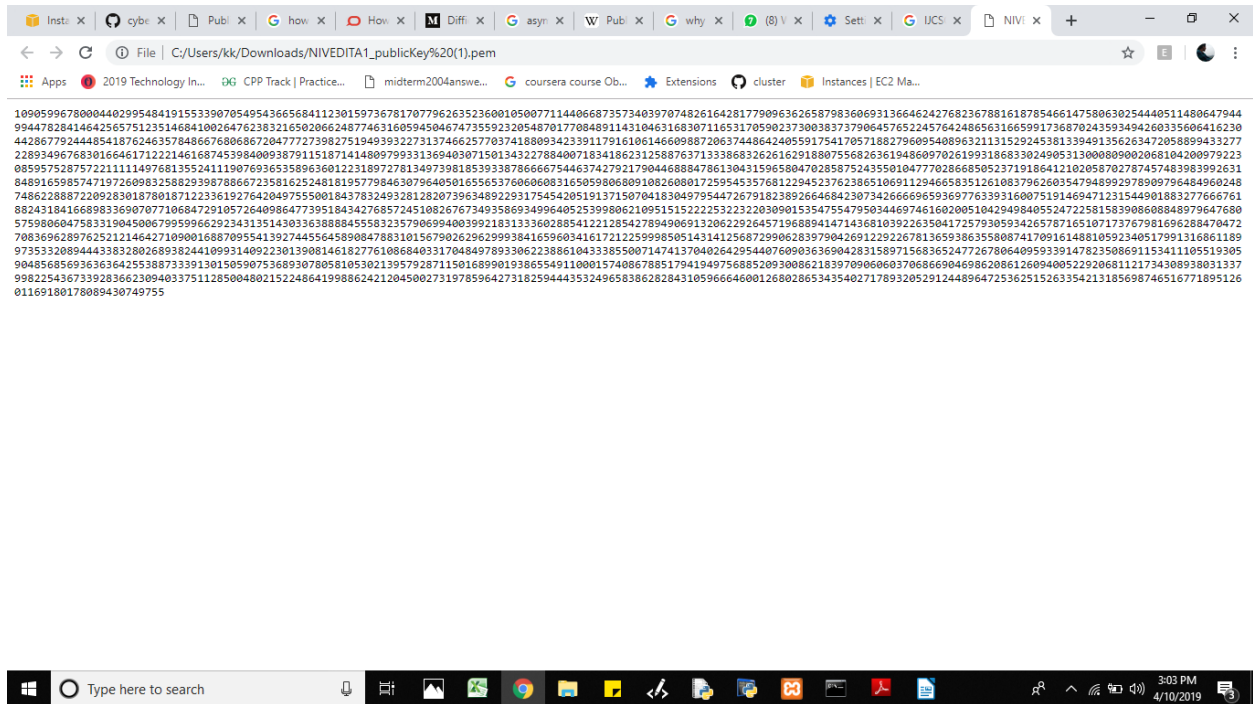
The Files Stored on the Cloud

Public Key Link	Uploader's Username
Click Here to Download the Public Key	Ast97
Click Here to Download the Public Key	Hina
Click Here to Download the Public Key	Astha
Click Here to Download the Public Key	nis
Click Here to Download the Public Key	Nivedita
Click Here to Download the Public Key	vitrnivi
Click Here to Download the Public Key	Aditi

Type here to search

11:23 PM 4/7/2019

The public Key file:



The Tkinter GUI:

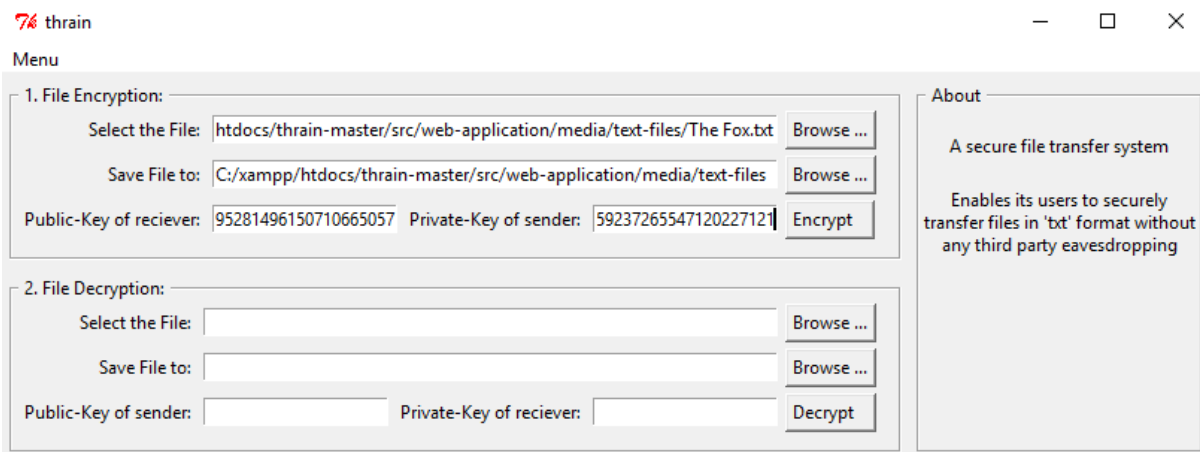
The screenshot shows a Tkinter window titled "7x thrain" with standard window controls (minimize, maximize, close). A "Menu" button is located in the top-left corner. The main content area is divided into two sections: "1. File Encryption:" and "2. File Decryption:". Each section contains two text input fields for file selection, two text input fields for public and private keys, and a button ("Browse ..." for file selection, "Encrypt" or "Decrypt" for the operation). An "About" panel on the right side of the window contains the text: "A secure file transfer system" and "Enables its users to securely transfer files in 'txt' format without any third party eavesdropping".

Now, suppose the sender is the user “Nivedita” and the receiver is the user “Astha”.

File to Encrypt:

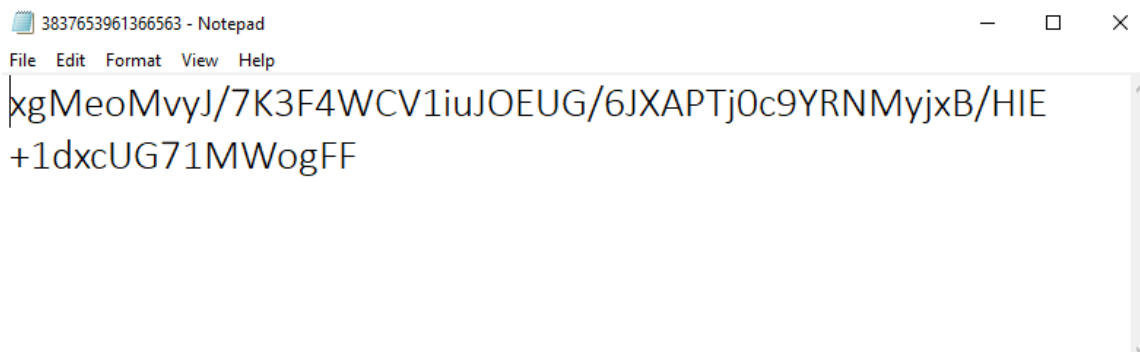
The screenshot shows a Notepad window titled "The Fox - Notepad" with a menu bar (File, Edit, Format, View, Help). The text area contains the sentence "The quick brown fox jumps over the lazy dog" followed by a cursor. A vertical scrollbar is visible on the right side of the text area.

Encrypt File: Public key of receiver i.e. “Astha” and private key of sender i.e. “Nivedita ” is required for encryption

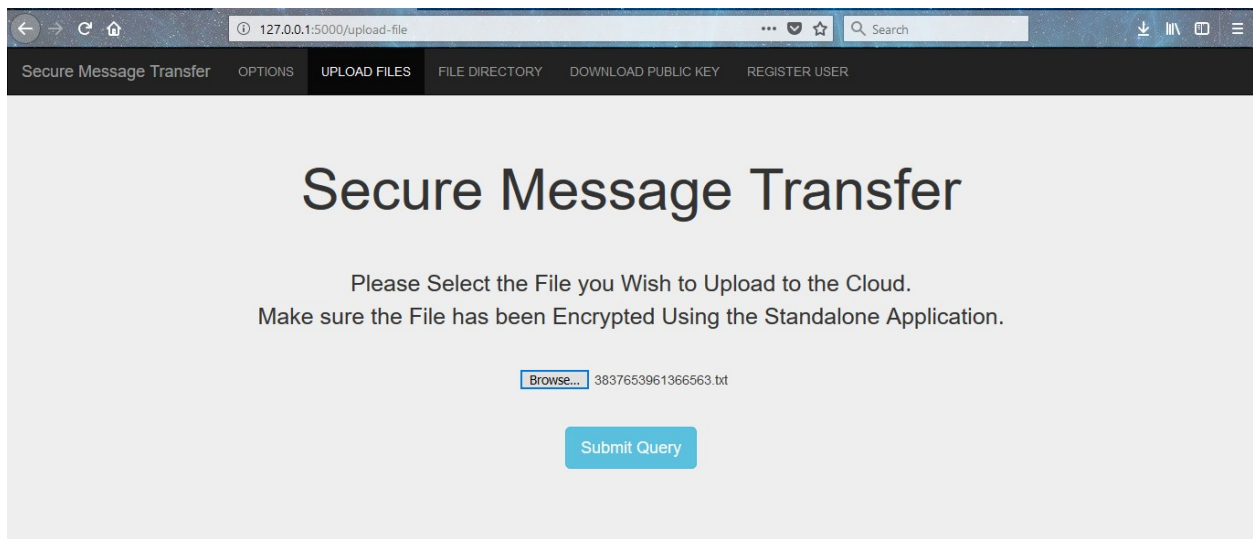
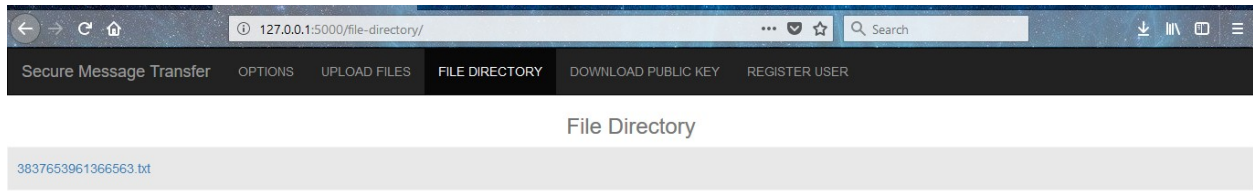


The screenshot shows a web application titled "thrain" with a menu bar. The main interface is divided into two sections: "1. File Encryption:" and "2. File Decryption:". The "File Encryption" section has fields for "Select the File:" (with a file path and a "Browse ..." button), "Save File to:" (with a file path and a "Browse ..." button), "Public-Key of reciever:" (with a key value), "Private-Key of sender:" (with a key value), and an "Encrypt" button. The "File Decryption" section has fields for "Select the File:" (with a "Browse ..." button), "Save File to:" (with a "Browse ..." button), "Public-Key of sender:" (with a "Browse ..." button), "Private-Key of reciever:" (with a "Browse ..." button), and a "Decrypt" button. On the right side, there is an "About" section with the text: "A secure file transfer system" and "Enables its users to securely transfer files in 'txt' format without any third party eavesdropping".

The Encrypted File:



The screenshot shows a Notepad window titled "3837653961366563 - Notepad". The text content is:
xgMeoMvyJ/7K3F4WCV1iuJ0EUG/6JXAPTj0c9YRNMvjxB/HIE
+1dxcUG71MWogFF



The encrypted “The Fox.txt” is successfully uploaded on cloud:

In this case the user has uploaded his file on the cloud. Now he has the following three options:

He can either get the list of all the different files that are present on the cloud.

He can also navigate back to the home page of this website.

He may even choose to upload another file on the cloud.

To decrypt the encrypted file: Public key of sender i.e. “Nivedita” and private key of receiver i.e. “AsthA” is required for decryption

76 thrain

Menu

1. File Encryption:

Select the File:

Browse ...

Save File to:

Browse ...

Public-Key of reciever:

Private-Key of sender:

Encrypt

2. File Decryption:

Select the File:

Browse ...

Save File to:

Browse ...

Public-Key of sender:

Private-Key of reciever:

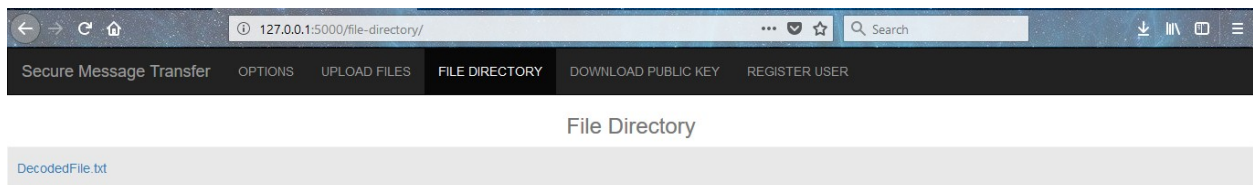
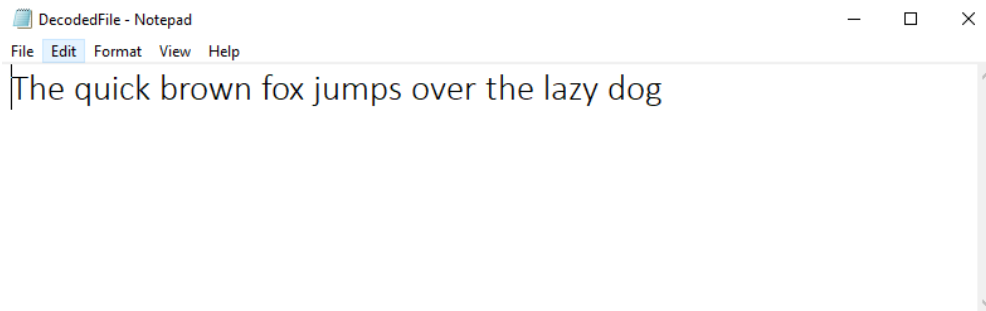
Decrypt

About

A secure file transfer system

Enables its users to securely transfer files in 'txt' format without any third party eavesdropping

The Decrypted File:



SOURCE CODE SNIPPETS

DIFFIE HELLMAN

```

prime = 0xFFFFFFFFFFFFFFFFC90FDAA22168C234C4C6628B80DC1CD129024E088A67CC74020BBEA63B139B22514A08798E3404DDEF9519B3CD3A431B302B0A6DF25F14374FE1356D6D51C245E485B576625E7
#GLOBAL PRIMITIVE ROOT
generator = 2
key_length = 600

...

***** DIFFIE HELLMAN KEY EXCHANGE PROTOCOL *****
...

def generate_private_key(length):
    _rand = 0
    _bytes = length // 8 + 8
    #Generate a random private key such that it's less than the prime number
    while (_rand.bit_length() < length):
        #TODO: Can use Crypto library hash functions
        hex_key = binascii.b2a_hex(os.urandom(_bytes))
        #Convert to denary format
        _rand = int(hex_key.encode('hex'),16)

    #Update Object
    private_key = _rand
    return private_key

#Public key = primitive root ^ private key % prime
def generate_public_key(private_key):
    public_key = pow(generator, private_key, prime)
    return public_key

#Secret key = public key ^ private key % q
def generate_secret(private_key, public_key):
    #Formula
    secret = pow(long(public_key), long(private_key), prime)
    try:
        secret_bytes = secret.to_bytes(
            shared_secret.bit_length() // 8 + 1, byteorder="big")
    except AttributeError:
        secret_bytes = str(secret)

    #Generate hash key using SHA256
    key = hashlib.sha256()
    key.update(bytes(secret_bytes))
    secretKey = key.hexdigest()
    return secretKey

```

AES

```

class AESCipher(object):

    def __init__(self, key):
        self.key = key
        #self.key = hashlib.sha256(key.encode()).digest()

    def encrypt(self, message):
        """
        It is assumed that you use Python 3.0+
        , so plaintext's type must be str type(== unicode).
        """
        message = message.encode()
        raw = pad(message)
        cipher = AES.new(self.key, AES.MODE_CBC, iv())
        enc = cipher.encrypt(raw)
        return base64.b64encode(enc).decode('utf-8')

    def decrypt(self, enc):
        enc = base64.b64decode(enc)
        cipher = AES.new(self.key, AES.MODE_CBC, iv())
        dec = cipher.decrypt(enc)
        return unpad(dec).decode('utf-8')

```

6.0 CONCLUSION

The proposed project aims to address the problem of secure file storage on the cloud. we study about cloud computing its advantages and disadvantages. Users can store their data over the cloud. So the main issue in cloud environment is to provide security to the data so that an unauthorized user cannot use or modify the data stored in the cloud server. In order to provide the security homomorphic encryption scheme is used with Diffie -Hellman algorithm. The Diffie-Hellman algorithm is used for secure channel establishment and for mutual authentication.

This method is a basic implementation of the proposed methodology that can be improvised and customized according to the needs. It proposes to use encryption and Diffie Hellman to provide double layer of security to the files that are stored on the cloud. Diffie Hellman which is used for exchanging the keys and is used for sharing the key and that key is used for encryption and decryption purpose.

The aim was to provide the storage of data in cloud securely which is achieved and to provide a strong key for AES encryption and decryption, SHA-256 is used. This will enhance the security and performance of cloud computing during network attacks. Cloud needs a high performance as well as security because the data on cloud is stored at some far place. A new come up is built by the integration of authentication and Diffie-Hellman algorithm.

7.0 REFERENCES

- [1] Anitha Patil¹, P Pillai HOC College of Engineering and Technology, India,”
Enhanced-Elliptic Curve Diffie Hellman Algorithm for Secure Data Storage in Multi Cloud Environment”, International Journal of Intelligent Engineering and Systems, Volume-184, January-2018
- [2] S. Grace Sophia and S. Prabakeran, Efficient and Secure Data Sharing Using AES and Diffie Hellman Key Exchange Algorithm in cloud, Middle-East Journal of Scientific Research 2016
- [3] Security Algorithms for Cloud Computing International Conference on Computational Modeling and Security (CMS 2016). By Akashdeep Bhardwaja, Vinay Avasthi, Hanumat Sastry
- [4] Shaheed Udham Singh College of Engg, Tangori, Mohali,”Providing Data Security in Cloud Computing Environment using Diffie-Hellman and HMAC”, IJCSC, Volume 6 – March-2015.