IIT-Hyderabad

Project Document
[*Secure-Cloud*]

Computer Network 2
(CS3543)

Project Name	Secure-Cloud	
Date	*2019-03-31*	
Author	*Om Sitapara CS16BTECH11036*	*Shubham Kumar ES16BTECH11028*

Contents

1 SW Development Plan

1.1 Project Overview

1.1.1 Objective and Project Scope

Objective:

The main objective of this project is to store files on cloud server securely. The users can upload, download or share files on the cloud server. Also the server provides some functionality as ls command. Library crypto++ is used for all security tasks.

Scope:

The transfer of the files happens in an encrypted manner where the encryption of message is done using AES_CBC encryption. The AES symmetric key is generated using Diffie-Hellman key exchange. The server stores the user-name password mapping of users in MD5 hash format. Once the handshake is performed the users can perform several request to server as HTTP:

1) *CREATE <username> <password> : This request creates a new user on server where the username and password on server side will be stored as MD5. Once the user is created a directory for that user is created.*
2) *LOGIN <username> <password> : This request is for login for existing user.*
3) *DOWNLOAD <filename> : This command downloads the specified filename from the server if it exists.*
4) *UPLOAD <filename> <filesize> : This command uploads the file on the server if the file size is less than the users allowed space on server.*
5) *DELETE <filename> : This deletes the file from the server.*
6) *DELETE_USER : This command deletes the current logged in user*
7) *SHARE <filename> <user_to_share> : This command shares the filename to the other user given.*
8) *LOGOUT : To finish the session*
9) *LS : Returns all the files owned by the user and shared with that user.*
10) *RUN <filename> <command to compile> < command to execute>: This command runs the given filename on the server.*
11) *VERIFY <filename> : Checks the integrity of that file on the server.*

Major review items

Handshake Protocol : AES key generation using DH and verification
Encryption and Decryption : Encrypt and Decrypt the packets using AES.
Files Management : Storing files and their original hash with access details.
User Account Management : Storing username and passwords
Testing : Catch.hpp and automated testing using CircleCI

1.2 Assumptions, Dependencies and Constraints

Item	Assumptions, Dependencies and Constraints	Remarks
1.	Crypto++ : For all security tasks (v8)	Dependency
2.	Server and Clients to be on same LAN.	Constraint
3.	A catch.hpp file for testing	Dependency
4.	There is no backhand for the server so it is assumed that server is always running	Assumption

1.3 Roles and Responsibilities

Student Name	Roles and Responsibilities	
Om Sitapara cs16b36 Shubham Kumar es16b28	**Developer** **Software Requirements Analysis** Verifying requirements and performing analysis on requirements;	
Om Sitapara cs16b36 Shubham Kumar es16b28	**Developer** **Software Architecture -**Mapping the requirements into Architecture	
Software Development	*CircleCi and Build*	**Om Sitapara**
	Google Cloud	**Om Sitapara**
	Version control	**Om Sitapara, Shubham Kumar**
	Deffie-Hellman	**Om Sitapara**
	AES Encryption Decryption and Verification	**Shubham Kumar**
	CREATE	**Om Sitapara**

	LOGIN	**Shubham Kumar**
	DOWNLOAD	**Om Sitapara**
	UPLOAD	**Shubham Kumar**
	SHARE	**Om Sitapara, Shubham**
	DELETE	**Shubham Kumar**
	LOGOUT	**Shubham Kumar**
	RUN	**Om Sitapara, Shubham Kumar**
	LS	**Shubham Kumar**
	VERIFY	**Om Sitapara**

1.4 Development Plan

1.4.1 Development Schedule

Estimated Project Period	25/03/2019 - 30/04/2019
Project Team Size	2
Estimated Man Months	2

Milestone	**1st Review**	**Final Review**
Planned Schedule	2-April-2019	During final exam week.

1.4.2 Development Environment

Item	Development Environment	Remarks
Program Languages	*C++*	*Follow the OOP design rule*
Compiler, Build	*g++ v11*	*A new version is expected if a chip is changed* *Specify compiler version.*
Target Kernel	*LINUX 4.1.0 & above*	
Word Processor for Document Creation	*Google Docs.*	

Configuration Management	*Github(version control)*
	CircleCI(build and testing)	

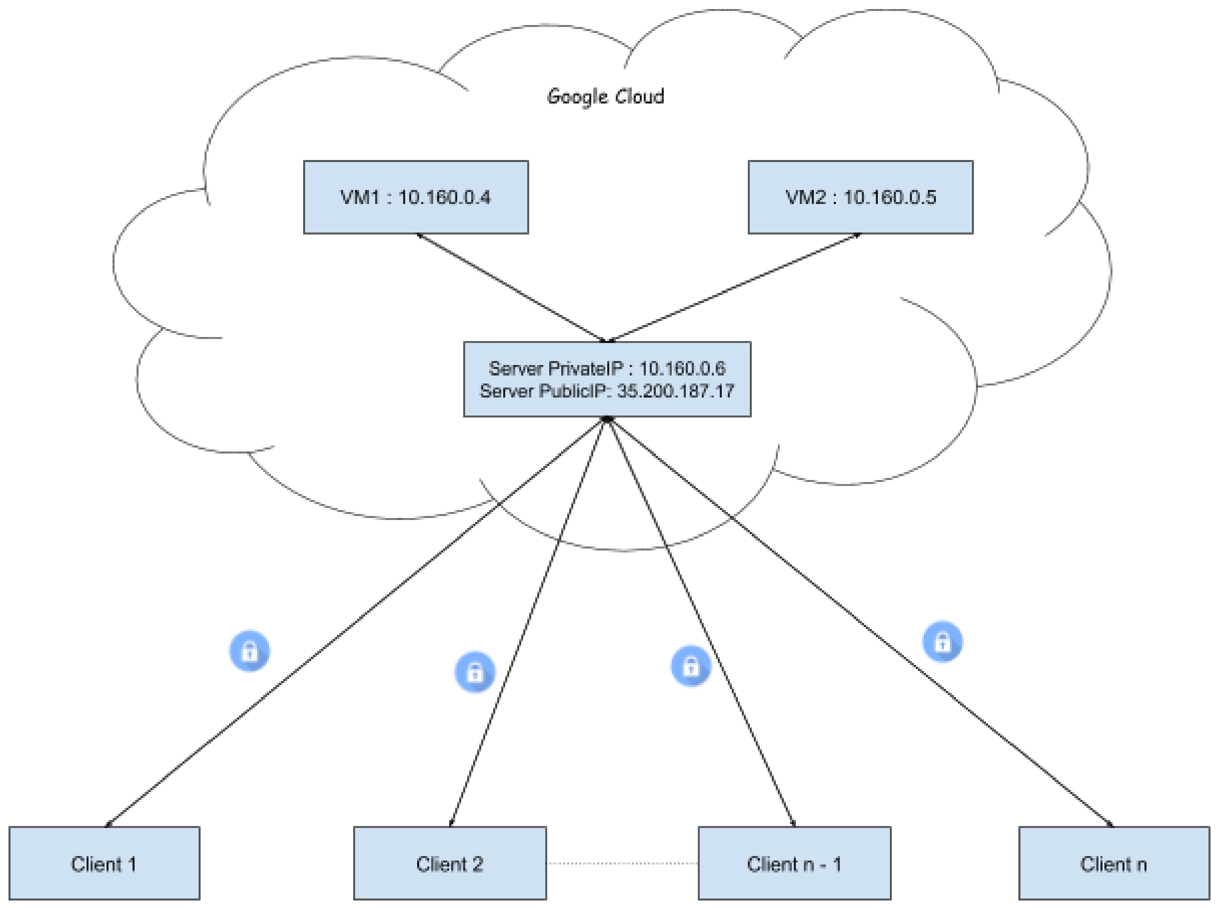
2 SW Requirements Specification

2.1 Major Functional Requirements

No	Requirement Id	Function Requirement Name	Description
1	*1*	*Deffie-hellman*	To generate symmetric AES key on both client and server
2	2	Sha256Digest	To generate master from pre-master secret.
3	3	UtilsFunction	To properly convert one form of data to other form for transfer via tcp.
4	4	Commands	To properly execute and process all the client request and saving the files on directory.

3 SW High & Detailed Level Design

3.1 Overall Architecture



3.2 SW System Operation Design

Represent the SW system Operation Design using Overall Class Diagram

3.2.1 {DesignID} Structure Diagram

3.2.1.1 {Class 'n'} Component Design

3.2.1.1.1 File Description

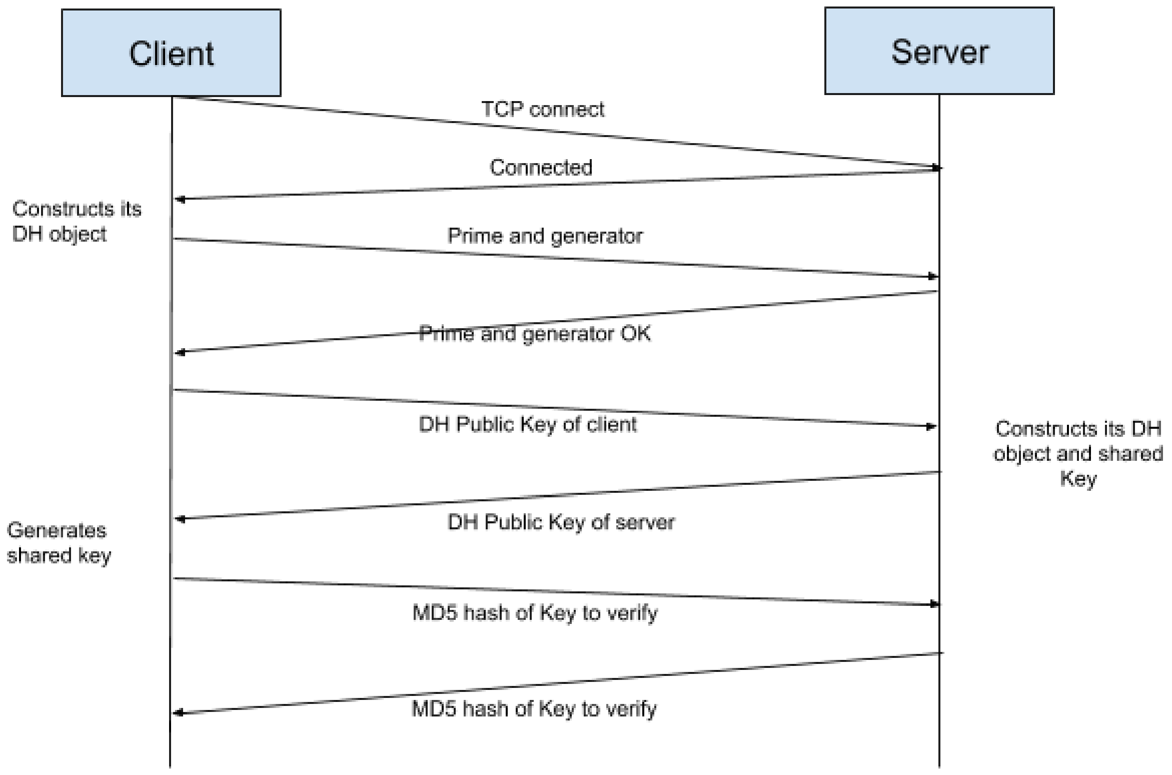
Describe the functions of the corresponding block.

Component	File	Description
Handshake	dhaes.hpp	This class has functions and data structure for the deffie-hellman key exchange.
Encryption	utils.hpp	This class has the functions for encryption and decryption of data
server	server.cpp	This file has the code for the server.
client	client.cpp	This file has the code for the client part
vm	vm.cpp	This file has the code for the vm which computes the files given by server.

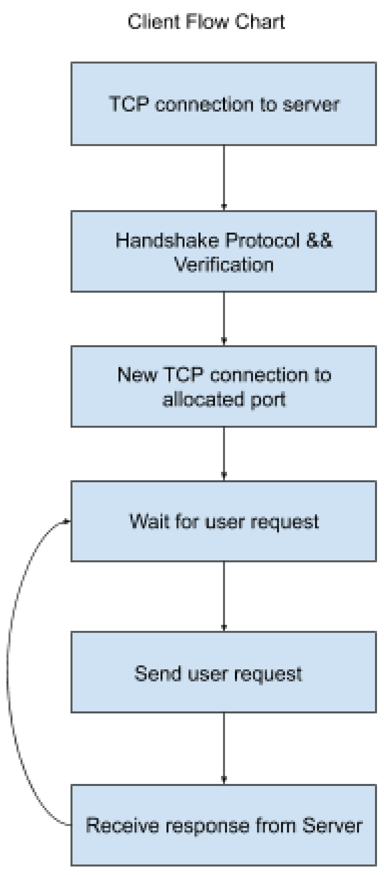
testing	utilsTest.cpp, dhtest.cpp	This are the files which performs the unit testing on the classes.

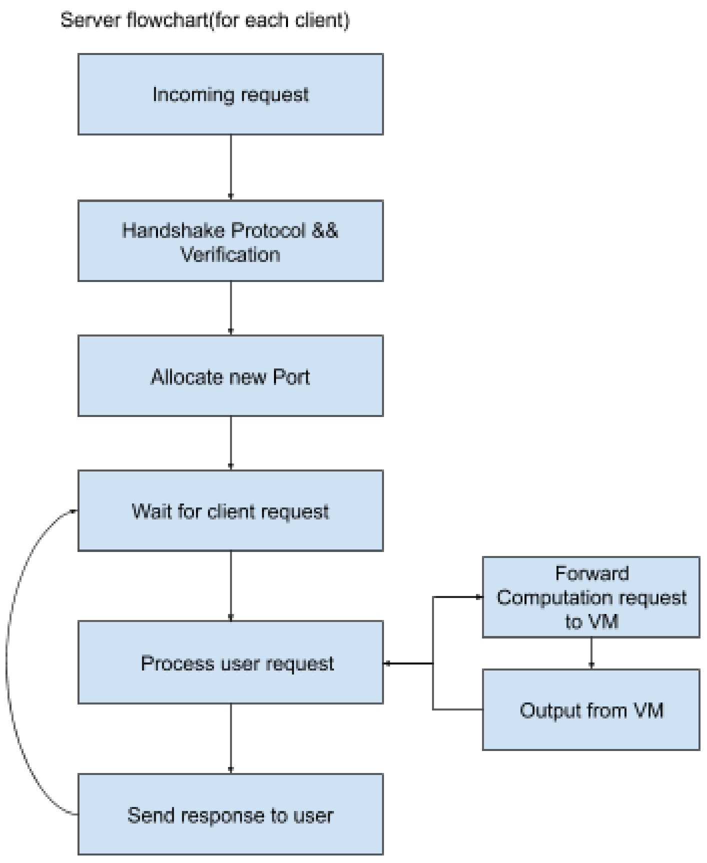
3.2.1.1.2 Sequence Diagram

Handshake Protocol :



Overall Flow charts :


Client Flow Chart
TCP connection to server
Handshake Protocol && Verification
New TCP connection to allocated port
Wait for user request
Send user request
Receive response from Server



3.3 SW Code Structure

Describe the code structure of the block. Insert a drawing or table that represents mapping modules to files. For

- **Mapping list of class and files (or folders)**

class name	File name (or folder name)
Deffie-Hellman	dhaes.hpp
Utils class	utils.hpp
server	server.cpp
client	client.cpp
vm	vm.cpp

3.4 Class vs. Function Mapping

Requirement ID	SW Design Elements		
	Component	Class/File	Function
1	rime, generator , public key generation	Diffie_Hellman	Default Constructor Diffie_Hellman()
2	rime, generator , public key generation with given parameters	Diffie_Hellman	Diffie_Hellman(Integer, Integer)
3	Symmetric key and its hash generation	ffie_Hellman	Agreefunc(SecByteBlock)
4	Getting the values of keys	ffie_Hellman	getPrime(), getGenerator(), getaesKey(), getpubKey(), getaesShaKey()
5	Converting keys to and from strings	utils	SecByteToString(), stringToSecByte(), IntegerToHexString(), StringToHexInteger()
6	nding MD5 hash of a key or a message string	utils	findMD5(SecByteBlock), findMD5(string)
7	Encrypting a message given the shared key and message length	utils	aesEncryption(SecByteBlock, char*, int)
8	Decrypting a cipher text given the shared key and its length	utils	aesDecryption(SecByteBlock, char*, int)
9	eating a socket and listening	server.cpp	main()
10	Accepting connections from clients	server.cpp	main() (using Select Activity)
11	Performing handshake using object of Diffie_Hellman class and detaching a thread for each client	server.cpp	main()
12	Sending new port to client, creating new socket , accepting connection to it and serving client requests	server.cpp	client_runner_th(client_soc)
13	arsing and processing all the client requests	server.cpp	parser_request(string, int, client_soc *)
14	creating TCP socket and connecting to the server	client.cpp	main()
15	tializing handshake by sending prime and generator to the server and then completing the rest of the handshake	client.cpp	reader(), writer()
16	aking input from the user on which operation to perform and sending corresponding requests to the server	client.cpp	main()
17	eceiving response from the server	client.cpp	main()
18	ating TCP socket for VMs that perform computation	vm.cpp	main()
19	cepting connections from the server to run programs	vm.cpp	accept_thread(int)

20	*Receiving Code file and commands to run from the server, executing the code and sending result file back to the server*	*vm.cpp*	*execute_func(int)*

4 SW Unit Test Report

Unit test report can be seen on circleCI :
https://circleci.com/gh/omsitapara23/Secure-Cloud/tree/master

5 SW Development Completion Report

5.1 Project Result Analysis

5.1.1 Development Results and Utilization

This is a combination of drive and cloud with some security. One can setup a personal cloud server inside a organization that can provide storage as well as computation power. Eg suppose for an organization like educational institute students can run their codes on this software if organization is running this secure cloud server and not have to rely on Google Cloud or AWS

5.1.2 Deliverables List

S.No	Executable Name	Description
1	*dhaes.hpp*	*Contains the code for Diffie_Hellman class.*
2	*utils.hpp*	*Contains the code for utils class.*
3	*server.cpp*	*Contains the code for the cloud server.*
4	*client.cpp*	*Contains the code for a single client.(Run on different terminals for multiple clients)*
5	*vm.cpp*	*Contains the code for VMs used for computation by the server on a client request.*
6	*dhtest.cpp*	*Contains tests written for Diffie_Hellman class functions.*
7	*utilsTest.cpp*	*Contains tests written for utils class functions.*
8	*catch.hpp*	*Contains code to run unit tests on functions.(Not coded by us)*

Guidelines to run code:

- You need to have lcrypto++ library installed to be able to compile and run the code. To install lcrytpo++ in ubuntu run the following commands:
  sudo apt-get update
  sudo apt-get install libcrypto++-dev libcrypto++-doc libcrypto++-utils
- For compiling server : g++ server.cpp -o s -std=c++11 -lpthread -lcrypto++
- Running server : ./s
- Now it will ask two ip for the vm which needs to be entered
- For compiling client : g++ client.cpp -o c -std=c++11 -lpthread -lcrypto++
- Running client : ./c
- Now it will ask the ip of the server which needs to be entered
- For compiling vm : g++ vm.cpp -o vm -std=c++11 -lpthread
- Running the vm : ./vm

Terminology / Abbreviations

Terminology / Abbreviations	***Description***
[Handshake]	*The procedure of establishing a shared key using Diffie hellman key exchange and then verifying it.*
[VM]	*Virtual Machine used for computation by the server.*
[Secure Channel]	*The server and the VMs reside inside a network with secure channels.*
[Authentic Channel]	*The channel connecting the server and the clients is authentic but not secure and we secure it by using encryption concepts.*

■ References

[1] crypto++ : https://www.cryptopp.com/wiki/Main_Page
[2]catch.hpp:https://raw.githubusercontent.com/catchorg/Catch2/master/single_include/catch2/catch.hpp
[3]Github Repo : https://github.com/omsitapara23/Secure-Cloud