Number Theory And Cryptography Report

Q) C1

A) Construct two programs: Server and Client to realize remote loginsystems with password authentication

Command line : ./a.out *server ip address (common for all C1 questions) for client*

Input Specification : It takes input username and password on client and send a login_req to server. The server maintains a login table(login_details.txt) which has usernames and passwords to process the request.

Output Specification: If the password and the username matches with any one of the login table, server gives the access to create a file named [Your Username].txt with the desired content. Otherwise it denies the access.

Work Flow : Client and Server creates a socket and communicate each other through the socket decriptor using Connect(), Send(), recv() functions

Sample Input 1:

```
Enter the username: praveen
Enter the password: 12345
        access granted(stdout)
Enter the content to make a file
        hello
Your file has been created(stdout)
```

Sample Output1: praveen.txt (with the content *hello*)

Sample Input 2 :

```
Enter the username: qwerty
Enter the password: 12345
```

Sample Output2: access denied

B- SHA-1)

Instead of plaintext storage utilize SHA-1 to protect the password on the server

Command line :

```
gcc server.c  sha1.c (server side)
gcc client.c (client side)
```

Input Specification : It takes input username and password on client and send a login_req to server. The server maintains a login table(login_details.txt) which has usernames and passwords in the hashing form.

Output Specification: If the password and the username matches with any one of the login table, server gives the access to create a file named [Your Username].txt with the desired content. Otherwise it denies the access.

Work Flow : Client and Server creates a socket and communicate each other through the socket decriptor using Connect(), Send(), recv() functions. SHA-1 algorithm implemented on server side receives the password from the client and converts it into the hashing form

Sample Input 1:

```
Enter the username: praveen
Enter the password: 12345
        access granted(stdout)
Enter the content to make a file
        hello
Your file has been created(stdout)
```

Sample Output1: praveen.txt (with the content *hello*)

Sample Input 2 :

```
Enter the username: qwerty
Enter the password: 12345
```

Sample Output2: access denied

B- AES)

Instead of plaintext communication utilize AES

Command line :

```
gcc server.c  sha1.c  aes.c aesni.c (server side)
gcc client.c  aes.c aesni.c(client side)
```

Input Specification : It takes input username and password on client and send a login_req to server. The server maintains a login table(login_details.txt) which has usernames and passwords in the hashing form.

Output Specification: If the password and the username matches with any one of the login table, server gives the access to create a file named [Your Username].txt with the desired content. Otherwise it denies the access.

Work Flow : Client and Server creates a socket and communicate each other through the socket decriptor using Connect(), Send(), recv() functions. AES algorithm encrypts and decrypts the data communicating into 128 bit blocks using 128 or 192 or 256 bit key which is a shared one

Sample Input 1:

```
Enter the username: praveen
Enter the password: 12345
        access granted(stdout)
Enter the content to make a file
        hello
Your file has been created(stdout)
```

Sample Output1: praveen.txt (with the content *hello*)

Sample Input 2 :

```
Enter the username: qwerty
Enter the password: 12345
```

Sample Output2: access denied

B- Key agreement)

Diffie–Hellman key exchange

Implement key agreement algorithm in your system

Command line : gcc server.c sha1.c aes.c aesni.c (server side)
gcc client.c aes.c aesni.c(client side)

Input Specification : It takes input username and password on client and send a login_req to server. The server maintains a login table(login_details.txt) which has usernames and passwords in the hashing form.

Output Specification: If the password and the username matches with any one of the login table, server gives the access to create a file named [Your Username].txt with the desired content. Otherwise it denies the access.

Work Flow : Client and Server creates a socket and communicate each other through the socket decriptor using Connect(), Send(), recv() functions. AES algorithm encrypts and decrypts the data communicating into 128 bit blocks using 128 or 192 or 256 bit key which is a shared one
But , the key is generated by Diffie-Hellman algorithm using generator and prime number shared on both sides , through which key can be generated without sharing it

Sample Input 1: Enter the username: praveen
Enter the password: 12345
access granted(stdout)
Enter the content to make a file
hello
Your file has been created(stdout)

Sample Output1: praveen.txt (with the content *hello*)

Sample Input 2 : Enter the username: qwerty
Enter the password: 12345

Sample Output2: access denied

Group Problem

P-7)

Implement double DES only, *no need to check its vulnerability against the attacks (after modification)*

Command line : gcc code_P7.c des.c

Input Specification : Enter data to be encrypted

Output Specification: Gives decrypted data that is already encrypted

Work Flow : Double DES uses multiple encryption. It encrypts plain text with two 56-bit keys and decrypts with that two keys in two stages.

Encryption is given as
C = E k2 [E k1 [P]] , k1,k2 are keys ; E- enc, P-PT
Decryption is given as
P = D k1 [D k2 [C]] , C-CT, D- decryption

Double DES uses DES twice with different keys

DES encrypts 64 bit block of plaintext. The encryption algorithm uses a 56-bit key. Encryption is done through a combination of substitution & permutation operations

Encryption is made of two permutations and 16 Feistel rounds.
Each round uses a 48 bit round key.
64 bit plaintext passes through an Initial permutation function.
The permuted data will go through 16 rounds of the same function.
The 56 bit key is passed through a permutation function.
For each round a subkey is produced.
Its produced by the combination of left circular shift and permutation.
The permutation function is the same for each round.
But different subkeys are produced.

Initial and Final permutations

Each of these P-Boxes take 64 bit input and permutes them according to a predefined rule.
These permutations are keyless and inverses of each other.
These P-Boxes have no cryptographic significance.

DES uses 16 rounds. Each round of DES is a Feistel cipher.

The left and right half of each 64 bit intermediate value are treated as separate 32 bit quantities.

Overall processing at each round
L i = R i-1
R i = L i-1 (XOR) F (R i-1 , K i)
Key K i is 48 bits.
The R input is 32 bits.
The R input is first expanded to 48 bits by using expansion permutation table.
It involves duplication of 16 bits of 32 bit Right half .

DES function applies a 48 bit key to the rightmost 32 bits to produce a 32 bit output.

This function is made up of 4 components

- An expansion P-Box
- A whitener
- A group of S- Boxes
- A straight P-Box

Expansion P-Box

We need to expand 32 bit right half of data to operate it with a 48 bit key.

Whitener (XOR)

DES uses it after the expansion permutation.
The inputs are the expanded right half and 48 bit round key.

S- Boxes

DES uses 8 S-Boxes each of which takes a 6 bit input and 4 bit output.
Each S- box is having 4 rows and 16 columns.
The 48 bit output from the whitener is divided into eight 6 bit chunks.
Each of these will be fed into the box, and the result will be 4 bits.

The first and last bits of the input identify the row and the middle 4 bits identify the column.
If the S- box consist of decimal value it will be converted to its binary equivalent.

Key is generated using Parity drop, Circular left shift and compression permutation.

Sample Input 1: Praveen

Sample Output1: Praveen

Sample Input 2 : NIT

Sample Output2: NIT