GUJARAT TECHNOLOGICAL UNIVERSITY

Master in Computer Application (Integrated)

Year IV – (Semester-VIII) (W.E.F. Dec 2016)

Subject Name: Network Security
Subject Code: 4480603

Indicative Practical List

Instructions:

All of the programs are to be implemented as stand-alone programs, with appropriate routines for encryption and decryption, etc. In all of the below mentioned programs, the input is read from keyboard. It is expected that students should also know how to implement the same programs in which input is read from a file or given as command line argument instead of keyboard. Similarly, students should know how to store the output in a separate file, apart from displaying the output on screen. In all such programs where a popular algorithm is to be used for a particular task like DES for encryption or MD5 for hashing, the existing libraries/classes/packages related to security are to be used, depending upon which development environment students are using. It is not expected that students should actually implement or develop the code of the given popular algorithm itself.

The programs may be implemented in Java/C++/C, as per the availability of the development environment in the host institute.

ASSESSMENT SCHEME

1. For programs related to confidentiality/encryption:

1	Proper execution of the program/generality (program executes properly for any valid data)	**15**
2	Display both the entered plaintext and cipher-text on screen.	**3**
3	Decrypt the cipher-text and display the decrypted contents on screen/in file (as per problem definition).	**3**
4	Usage of Appropriate Package/Methods/Library	**3**
5.	Viva Voce	**6**

2. For programs related to Message Digest:

1	Proper execution of the program/generality (program executes properly for any valid data)	**15**
2	Display of calculated Message Digest in Hex.	**3**
3	Display of Message Digest Size in bits	**3**
4	Usage of Appropriate Package/Methods/Library	**3**
5.	Viva Voce	**6**

PROGRAM LIST:

1. DES-Confidentiality

   Implement a program which encrypts a given text message entered from keyboard. Use DES algorithm for encryption. Display both the entered plaintext and cipher-text on screen. Decrypt the cipher-text and display the decrypted contents on screen. Use Appropriate package/Library for DES. Extend the above program to a menu based version where the user selects the operational mode (ECB/CBC/CFM) to be used and implement the program accordingly. Extend the above program to demonstrate the avalanche effect. This means, for the same key size, if two different pieces of data are encrypted, where each piece of data is of same size and varying in very less degree (say 1 bit), yet the corresponding ciphertext generated for both the data are significantly different. Similarly, avalanche effect should also be observed for same piece of data but two different key values where each key value is differing from the other key value by just 1 bit.

2. 3DES (Triple DES)-Confidentiality

   Implement a program which encrypts a given text message entered from keyboard. Use 3DES algorithm for encryption. Display both the entered plaintext and cipher-text on screen. Decrypt the cipher-text and display the decrypted contents on screen. Use Appropriate package/Library for 3DES.
   Extend the above program to a menu based version where the user selects the operational mode (ECB/CBC/CFM) to be used and implement the program accordingly. Implement both the 2 key and 3 key versions. Extend the above program to demonstrate the avalanche effect. This means, for the same key size, if two different pieces of data are encrypted, where each piece of data is of same size and varying in very less degree (say 1 bit), yet the corresponding ciphertext generated for both the data are significantly different. Similarly, avalanche effect should also be observed for same piece of data but two different key values where each key value is differing from the other key value by just 1 bit.

3. AES-Confidentiality

   Implement a program which encrypts a given text message entered from keyboard. Use AES algorithm for encryption. Display both the entered plaintext and cipher-text on screen. Decrypt the cipher-text and display the decrypted contents on screen. Use Appropriate package/Library for AES.
   Extend the above program to a menu based version where the user selects the operational mode (ECB/CBC/CFM) to be used and implement the program accordingly. Implement the program for all permissible key sizes. Extend the

above program to demonstrate the avalanche effect. This means, for the same key size, if two different pieces of data are encrypted, where each piece of data is of same size and varying in very less degree (say 1 bit), yet the corresponding ciphertext generated for both the data are significantly different. Similarly, avalanche effect should also be observed for same piece of data but two different key values where each key value is differing from the other key value by just 1 bit.

4. RSA-Confidentiality

   Implement a stand-alone sender receiver program in which Sender encrypts a given text message entered from keyboard using RSA public key of Receiver and sends it to Receiver. Receiver Decrypts the cipher-text using his own RSA Private key and display the decrypted contents on screen. Use Appropriate package/Library for RSA.

5. RSA-Authentication

   Implement a stand-alone sender receiver program in which Sender encrypts a given text message entered from keyboard using his own RSA Private Key and sends it to Receiver. Receiver Decrypts the cipher-text using sender's RSA Public Key and display the decrypted contents on screen. Use Appropriate package/Library for RSA.

   <u>On similar lines as of Program # 4 and 5, RSA based program which implements both confidentiality and authentication should be implemented. Also, programs related to application of RSA for implementing digital signature and key exchange, should be implemented</u>. Key exchange program should include/implement bulk encryption using conventional symmetric ciphers like DES/3DES/AES. The program should be implemented such that initially the symmetric key is exchanged using RSA and then the bulk encryption algorithm takes over.

6. RC4-Stream Cipher

   Implement a stand-alone sender receiver program in which Sender encrypts a given text message entered from keyboard in a byte by byte manner using RC4 Stream Cipher and transmits it to receiver. Receiver Decrypts the cipher-text displays the decrypted contents on screen. Use Appropriate package/Library for RC4.

7. MD5-Message Digest

   Implement a program which applies MD5 hashing on a given text message entered from keyboard. Display the hashed contents on screen and determine the size (in bits) of the output. Use appropriate Library/Package for MD5. Extend the above program to demonstrate the avalanche effect. This means, if message digest is applied for two different pieces of data, each of same size and differing by only 1 bit (say LSB), yet the corresponding message digest generated for each data are significantly different.

8. SHA-1-Message Digest

   Implement a program which applies SHA-1 hashing on a given text message entered from keyboard. Display the hashed contents on screen and determine the size (in bits) of the output. Use appropriate Library/Package for SHA-1. Extend the above program to demonstrate the avalanche effect. This means, if message digest is applied for two different pieces of data, each of same size and differing by only 1 bit (say LSB), yet the corresponding message digest generated for each data are significantly different.

9. SHA-512-Message Digest

   Implement a program which applies SHA-512 hashing on a given text message entered from keyboard. Display the hashed contents on screen and determine the size (in bits) of the output. Use appropriate Library/Package for SHA-512. Extend the above program to demonstrate the avalanche effect. This means, if message digest is applied for two different pieces of data, each of same size and differing by only 1 bit (say LSB), yet the corresponding message digest generated for each data are significantly different.

10. SHA-1-PASSWORD

    Implement a program in which the user enters a password from keyboard. The program applies SHA-1 hash on the entered password and stores it in a file. The program displays a message “Password hash stored”. After this message, program prompts the user to re-enter the same password. When the user enters the password, the program computes the SHA-1 hash of the entered password. It then compares the SHA-1 hash of the entered password with the stored hash. If it is same, it displays the message “Successful” else displays a message “Error”.

On similar lines, message digest based programs for authenticating password can be implemented using MD5 and SHA-512 also.

11. HMAC-MD5

Implement a program which computes HMAC-MD5 hash for a given text message entered from keyboard. Display the contents and the size of the hash code. Now, compute the hash using only MD5 for the same text and display the contents and size of the hash. What are your observations for both the cases? Use Appropriate package/Library for HMAC-MD5 and MD5.

On similar lines, programs can be implemented for HMAC-SHA1 AND HMAC-SHA512 also.

12. DIFFIE HELLMEN KEY EXCHANGE:

Implement a sender receiver program where the sender sends the necessary parameter values to receiver. Using these values, the receiver generates the same key value as that of the sender. Both sender and receiver should display the generated values on screen. Use Diffi-Hellmen Key exchange Method. Use Appropriate package/Library for Diffie-Hellmen.

13. PASSWORD STRENGTH:

Implement a program in which the user enters a password. The program should categorize the entered password as secure or in-secure based upon the following conditions:

a) The minimum no. of characters should be 8
b) There should be at-least two upper case letters.
c) There should be at-least two digits.
d) No letters other then alphanumeric are allowed.

14. BRUTE FORCE:

Implement a program where a user enters a 8 character password. This password is hashed using MD5 and stored. Assume that the plaintext password consisting only of alphabetic CAPITAL CASE character only is allowed. Now, implement a routine TRY() which keeps on generating random 8 character strings consisting of CAPITAL CASE only. Whenever a sample value is generated, its MD5 is generated and compared with the stored MD5. If both the MD5 match, the TRY() routine displays a message “ CRACKED!” and displays the total no. of trials executed including the current trial, terminates and exits the program. If the generated sample’s MD5 does not match the stored MD5, the TRY() routine displays the message “MISSED” and displays the current trial number and continues further.

15. SIMPLE CHARACTER BASED CIPHERS-COLUMNAR CIPHER:

Implement a program which accepts 64 character (with no white space, only alphabets) text as input from keyboard. There is a routine ENCRYPT() which re-arranges the text as 8 x 8 matrix with first 8 characters stored as first row, next eight characters as second row and so on. Now, it transmits this matrix in a column-wise manner to another routine DECRYPT. The DECRYPT routine stores the incoming data as 8x 8 matrix , storing first eight characters as first column, second eight characters as second column and so on. Now, it displays all the stored rows, starting from first row up-to the eighth row on screen. This should match with the original data keyed in by the user.

16. SIMPLE CHARACTER BASED CIPHERS-CAESAR CIPHER:

Implement a program where the user enters a lowercase alphabetical text message from keyboard. There is a routine ENCRYPT() which shifts each of the letters of the text message to next character. Eg. b is encrypted to c, c to d and so on. z is encrypted back to a. There is another routine DECRYPT() which accepts the cipher-text as input and performs the reverse operation. i.e. c is decrypted to b, k to j and so on. Compare the contents of the plaintext and the decrypted ciphertext, which should turn-out to be same.

-x-x-x-x-x-