EN4720: Security in Cyber-Physical Systems
Exercise — Encryption and Hashing

Name: Thalagala B. P.
Index No: 180631J

March 13, 2023

This is an individual exercise!
Due Date: 21 April 2022 by 11.59 PM

Problem 1 - Symmetric Encryption & Key Exchange

Alice and Bob are using the Diffie-Hellman protocol as shown in Figure 1. They want to extend their key exchange mechanism to include their friend Charlie.

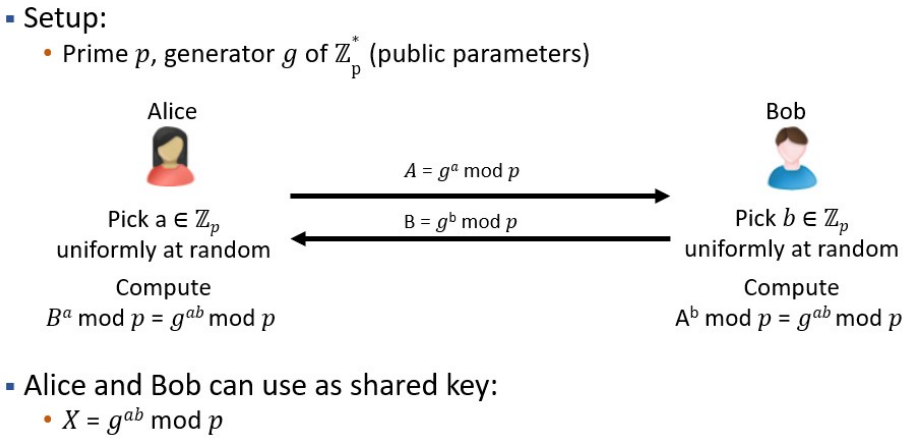

Figure 1: Diffie Hellman Protocol

1. Design an extension of the Diffie-Hellman protocol to allow secure key exchange between the three parties.

   Your answer here

2. Alice, Bob, and Charlie are planning to use $(p, g) = (23, 9)$ to agree on a key for a Caesar cipher. Assume that the Caesar cipher is right shifting the number of letters equal to the key to produce the ciphertext (e.g., if the key is 5 and the plaintext letter is U, it will be replaced by Z in the ciphertext). Assuming 4, 7, and 13 to be secret exponents of Alice, Bob, and Charlie respectively, find the common key that they need to use in Caesar's cipher.

   Your answer here

3. Derive the encrypted message for the message "Hello" using the above encryption key. Show all the major steps in deriving the answer.

   Your answer here

Problem 2 - Symmetric Encryption - Security Analysis

For this part of the exercise you are required to install CrypTool. CrypTool is a software tool that comes with several encryption techniques.

Instructions

1. Create a text file named **"sampletext.txt"** in your computer. Add the text "EN4720: This is a sample text file for encryption." to the text file and save it.
2. Open the **"sampletext.txt"** file with CrypTool software.
3. Encrypt your message with RC2 encryption following **Encrypt/Decrypt >Symmetric (modern) >RC2**. Choose key size of 8 bits and add some random hexadecimal key to encrypt your text file. (Add screenshots)
4. Run brute force analysis of the encrypted message by following **Analysis >Symmetric Encryption (modern) >RC2**. (Add screenshots)
5. Repeat steps 3,4 for the key sizes 16, 24, 64, and 128 bits.
6. Answer the questions given below.

Questions

1. Add screenshots requested in Step #3.

   Your answer here

2. Add screenshots requested in Step #4.

   Your answer here

3. Complete Table 1.

Table 1: Time taken to brute force

Key length (bits)	Time to brute force	Maximum number of keys
8		
16		
24		
64		
128		

4. What is the impact of key length on time to decrypt a message using brute force?

   Your answer here

5. What could be done to diminish the amount of time required to perform the attack?

   Your answer here

Problem 3 - Public Key Encryption

You are given that $n = 65$ and $\phi(n) = 48$ for a certain RSA scheme.

1. Calculate the private key d assuming the public key $e = 7$.
2. Show all major steps of encryption and decryption using textbook RSA for the message given below. Specifically, derive the ciphertext and show that the plaintext can be recovered at the receiver after decrypting.
   m = last digit of your index number (e.g., if the index number is 180234N, $m = 4$). If the last digital is 0 (zero) or 1 (one), use $m = 5$.
3. Find the candidates for the primes p, q that were used to generate this RSA scheme.
4. Suppose that c is the ciphertext that you find above for the message m. Show how an attacker can recover the plaintext message m from the ciphertext c assuming that he can request a decryption of a single ciphertext $c' \neq c$.

Problem 4 - Hash Functions

For this part of the exercise you are required to install Kali Linux. Kali Linux comes preinstalled with several hashing utilities. These include **md5sum**, **sha1sum**, **sha256sum**, and **sha512sum**. Follow the given instructions to complete the problem.

Instructions

1. Create a new directory and a new text file "**originalhashfile**" in the directory using **nano originalhashfile** command. Add the text "EN4720: Some text here" in the created file and save the file using the hotkey **CTRL+X** followed by **Y** and **Enter**.
2. Run the *ls* command in the directory to show the file (add screenshot) and run the *cat* command to show the contents of the created file (add screenshot).
3. calculate the hash digest of the **originalhashfile** using these hash algorithms: MD5, SHA1, SHA256, SHA512 (add screenshots). Add your observations in Table 2. You can use the following commands to get the results.
   - md5sum <*hashfile_name*>
   - sha1sum <*hashfile_name*>
   - sha256sum <*hashfile_name*>
   - sha512sum <*hashfile_name*>
4. Change the contents of the **originalhashfile** by adding your index at the end of the line ("EN4720: Some text here <*index_no*>") and save it as "**modifiedhashfile**" in the same directory.
5. Run the *ls* command in the directory to show the files (add screenshot) and run the *cat* command to show the contents of the modified file (add screenshot).
6. Calculate the hash digest of the **modifiedhashfile** using these hash algorithms: MD5, SHA1, SHA256, SHA512 (add screenshots). Add your observations in Table 3.
7. Answer the questions given below.

Questions

1. Add screenshots requested in Step #2.

   Your answer here

2. Complete Table 2. Add screenshots requested in Step #3 below the table.

Table 2: Hash digests of the original file

Hash Algorithm	**Hash of the Original File**
MD5	
SHA1	
SHA256	
SHA512	

3. Among the hash algorithms used, which one is the most cryptographically secure? Justify your answer.

   Your answer here

4. Why is MD5 not considered as a reliable hashing algorithm?

   Your answer here

5. Which of the algorithms fall under the category of SHA-2?

   Your answer here

6. Add screenshots requested in Step #5.

   Your answer here

7. Complete Table 3. Add screenshots requested in Step #6 below the table.

Table 3: Hash digests of the modified file

Hash Algorithm	**Hash of the Modified File**
MD5	
SHA1	
SHA256	
SHA512	

8. Comparing the output from one algorithm in Table 2 and Table 3, is there a difference between the hash digests? If yes, why?

   Your answer here