Homework 4

CSL7480: Cryptography

The assignment comprises 5 questions. Generate one Google Colaboratory file per question and a common report explaining the approach and results. Include the link to all the colab files in the report only.
Submit the report as a pdf file and name it as "[Roll_No]_[Name]_Assignment4.pdf"

Additional Details - File with different values to be used in Homework 4.

Preliminary:

RSA encryption

- Public Key is referred as (e, n)
- Private Key is referred as (d, n)
- Prime numbers are referred as p and q, and n = p.q
- "M" represents the original message and "x" represents the ciphertext.

PKCS1.5 Standard

- Let the hexadecimal representation of an integer 1 (in hex '01') be written as 01h.
- Initially the message "m" is converted into hash value "HASH" using one of the cryptographic hash functions.
- The code for the chosen hashing function and the length of the hash is stored as ASN.1.
- Finally, the hash value is formatted as "00h 01h ffh ffh ... ffh ffh 00h ASN.1 HASH", where the first two bytes are 00h and 01h, followed by ffh bytes (redundant bytes for padding). Then 00h determines the starting of the ASN.1 and hash value.Using padding the formatted message length is increased to be the same as of N.
- For example, a valid PKCS1.5 standard message is "00h 01h ffh ffh 00h 30h ..." where ... contains 48 bytes (=hex 30) of the hash of the original message.

Message Verification

In order to verify integrity of the message, two steps are followed:

- First it is checked that the message is compliant with PKCS1.5 standard
    - Initial byte is 00h followed by 01h.
    - Then it further looks for 00h byte
    - The next byte to 00h is considered as ASN.1
    - As per the information in ASN.1, the next few bytes are considered to be the hash.

- Next, the hash obtained above is verified against the hash of the original message.

Note - The verification process doesn't verify the number of redundant bits. Hence, messages formatted as "00h 01h ffh ... ffh 00h ASN.1 HASH **Garbage**" will also be verified.

Questions:

1. A message "m" is encrypted using textbook RSA algorithm with 3 different modulus but the same encryption exponents, correspondingly generating 3 ciphertexts. Now, using the three ciphertext and the associated public keys, design an algorithm to obtain the original message "m".

   The ciphertexts and their corresponding public keys are provided in the attached excel sheet. For instance - Ciphertext C1 is generated using e = K1[0] and n = K1[1].

   The report should clearly highlight the obtained message "m".

2. Follow the underlying steps
   a. Encrypt message "m" using the textbook RSA algorithm to generate ciphertext "x". And values for message "m", public encryption exponent "e" and the prime numbers "p" & "q" are mentioned in the Excel sheet for RSA algorithm.

   b. Determine the private key.
   c. Design the decryption oracle that accepts a ciphertext and then decrypts it using the private key to return the corresponding plaintext.

      Note - The ciphertext "x" can't be provided as an input to this function.

   Your task is to obtain the message "m" without passing "x" as input to the decryption oracle. Write proper code to show the results.

3. This question relates to the task of Digital Signature Verification using RSA. Please, follow the underlying steps:
   a. Generate RSA key pairs, where the public key encryption exponent "e" should be 3. And consider the message "m" to be your roll number (all small letters). For eg - "p21cs011".
   b. Digitally sign the message "m" using the private key with PKCS1.5 padding to obtain signed text "x" (Inbuilt libraries support automatic message conversion followed by encryption). You can use any cryptographic hashing function for the same.

Note - Private key can be used only once for signing the original message.

c. Design a server (or function) that accepts the ciphertext, authenticates the sender by decrypting the signed message using the public key and verifying the message (plaintext) as per preliminaries - “Message Verification”. The server returns a boolean value indicating the verification results. For instance - If the format is not valid, the server responds with “False” value. On the other hand, if the format is valid, the server returns “True”.

   That is, there exists a PKCS Padding oracle to tell you if the padding of a message is valid or not.

Your task is to generate another ciphertext that passes the PKCS padding oracle verification. Importantly, the direct use of both public & private keys is prohibited. The task should be accomplished using only the PKCS padding oracle.

4. Follow the underlying steps
   a. Encrypt message "m" using the textbook RSA algorithm to generate ciphertext “x”. And values for message “m”, public encryption exponent “e” and the prime numbers “p” & “q” are mentioned in the Excel sheet for RSA algorithm.
   b. Determine the private key.
   c. Design a parity oracle that accepts a ciphertext and decrypts it using the private key. The oracle then returns a boolean value to indicate the parity of the last bit of the plaintext. For instance, if the last bit is odd then the parity oracle returns “True” else “False”.

   Your task is to obtain the message "m" from ciphertext “x” using parity oracle only. Write proper code to show the results.

5. Refer to the Bleichenbacher's attack as mentioned in paper title "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1".
   Follow the underlying steps:
   a. Generate RSA key pairs, where the public key encryption exponent “e” should be 3. And consider the message “m” to be your roll number (all small letters). For eg - "p21cs011".

      p and q should be 128 bit prime numbers.

b. Encrypt the message “m” using the RSA algorithm with PKCS1.5 padding to obtain ciphertext “x” (Inbuilt libraries support automatic message conversion followed by encryption). You can use any cryptographic hashing function for the same.
c. Design a server (or function) that accepts the ciphertext, decrypts it using the private key and verifies the message (plaintext) has the first two bytes as 00h 01h. The server returns a boolean value indicating the verification results. For instance - If the first two bytes are anything other than 00h 01h, the server responds with “False” value. On the other hand, the server returns “True” value.

Your task is to decrypt the ciphertext “x” using the server defined in this question following the Bleichenbacher's attack.