ECE424: Network security

Homework 3
Due: 10/30/2024

Collaboration Policy. Each student must hand in their own answers. Use of partial or entire solutions obtained from others or online is strictly prohibited. However, discussing the problems with other fellow students or forming study groups is encouraged.

Submission Format. Please submit your solutions electronically through Canvas as a **single PDF file**. This file should include both your code and clear explanations of each step involved. Ensure to also provide the **outputs** generated by your codes. Additionally, include answers to any non-code related questions. It is crucial that your solutions are organized in a way that allows for easy review and grading by our team.

Grading. All the Roman Numerals items need to be answered with outputs. The grading will be based on the success outputs. The outputs should be screenshots including code panel and output panel.

1. *Auxiliary functions.* In this problem, you implement some functions that are needed in implementing the RSA algorithm in the next problem.

   (a) `exponentiation`: Implement a function that receives a message m, power e, and basis n, outputs m^e mod n. (Hint: Use the method explained in class for efficient exponentiation.)

      i. Verify the result with $m = 2000$, $e = 2020$, $n = 2023$.

   (b) `inverse_finder`: You need a function that receives a and n, such that $\gcd(a, n) = 1$, and outputs $b = a^{-1}$, i.e., b such that $ab \equiv 1 \pmod{n}$.

      i. Verify the result with $a = 2000$ and $n = 2023$.

2. *Implementing RSA* Write a code that implements RSA algorithm. Your code should have three different functions:

   (a) `RSA_key_generate`: Outputs both public and private key ($k_e = (e, n)$ and $k_d = (d, n)$). Input: e. Assume that input e is optional, such that if the user does not provide e, then one of the default values discussed in class will be selected.

   - The RSA key generation requires two large prime numbers p and q. Use existing random number generating functions and prime number verification functions of your coding language to generate two random prime integers between 0 and $2^{512} - 1$.
   - You need to verify whether e and $\phi(pq)$ are coprime as well as whether $e < \phi(pq)$. Your function `RSA_key_generate` should only output keys that satisfy these conditions.

i. Set $e = 3$, run `RSA_key_generate` 10 times. Report the generated keys.

(b) `RSA_encrypt`: Receives message m, which is between 0 and $n-1$, and $k_e = (e, n)$ and outputs ciphertext.

(c) `RSA_decrypt`: Receives ciphertext c, which is between 0 and $n-1$, and $k_d = (d, n)$ and recovers the message.

i. Set $e = 2^{16} + 1 = 65537$. Report your public key and private key.

ii. Verify your `RSA_encrypt`, `RSA_decrypt` with the above key and message 123456789. You need to show the decrypted message is the same as the original message.

iii. Verify your `RSA_encrypt`, `RSA_decrypt` with the above key and message 'hello world!' (case and punctuation sensitive). You need to show the decrypted message is the same as the original message.

3. *The Pohlig-Hellman algorithm.*

(a) Is 2 a primitive root mod 13? How about 3?

(b) Use the Pohlig-Hellman algorithm to solve

$$2^x \equiv 12 \pmod{13}.$$

(c) Use the result of part (b), to solve,

$$2^x \equiv 3 \pmod{13}.$$