Handout: RSA encoding/decoding

Introduction

In 1977, the RSA encryption algorithm[1] was created by Ron Rivest, Adi Shamir, and Leonard Adleman (R.S.A.) at MIT. The algorithm depends on having two "keys" – one is a public key and one is a private key – that are based on large prime numbers. The algorithm works as follows.

- Bob wants to send Alice a secret message.
  1. Alice shares her public key (n, e) with Bob but never shares her private key, d.
  2. Bob converts his message into an integer M with $0 \leq M < n$.
  3. Then, Bob encodes M using Alice's public key, e, to get the ciphertext, C.
  4. Finally, Bob sends the encoded message to Alice.
- Now, Alice wants to decode Bob's message, C.
  1. Alice decodes the message C using her private key, d, to recover the message M.
  2. Alice converts the integer M back to plain text characters and reads the message.
- Note that if someone, say Eve, knows the public key, (n, e), and the encoded message, C, Eve won't be able to decode the message without d, the private key that only Alice has.

The key breakthrough in the RSA cryptosystem is that Bob can encode a message using a publicly available key while knowing that no one can decode the message without having Alice's private key. (This is similar to Bob being able to lock a treasure chest with a key that anyone can have, but only Alice can open the treasure chest with her private key.)

The RSA algorithm depends heavily on the use of two large prime numbers, p, q, that are used to create $n = pq$. Currently, factoring a very, very large number into two extremely large primes is an incredibly difficult problem, which provides the security needed for the system to work.

Practice

1. Write a python program to encode and decode a given string using the RSA cipher with digraphs. You should have functions that do the following:
   (a) Convert a string into a list containing lists of digraphs. (Example: If we input "hello", the function should convert the string into `[[7, 4], [11, 11], [14, 23]]`.)
   (b) Convert a list containing a digraph `[a,b]` into an integer by creating the value $M = a \cdot 100 + b$.

[1] You can obtain their original paper at https://people.csail.mit.edu/rivest/Rsapaper.pdf

(c) Encode an integer using $C = M^e \bmod n$. **Since we will be using large integers later on, you will need to use the `pow` command in Python to exponentiate mod n. The syntax is: `pow(M,e,n)` gives M^e mod n.**

(d) Decode an integer using $M = C^d \bmod n$.

(e) Convert an integer M into a digraph by writing `[M / 100,M % 100]`.

(f) Convert a list containing lists of digraphs into a string. (Example: If we input `[[7, 4], [11, 11], [14, 23]]`, the function should return "hello".)

For the encoding, use the values $n = 1907 \cdot 2411 = 4597777$, $d = 1299277$, and $e = 63273$.

2. Submit the results of encoding and decoding several test strings. Also, I will e-mail the class an encoded message that you must decode and turn in.

3. Encode and email me a secret message. My public key will be

$$(n = 2659197123899, e = 583651243)$$