Proyecto

Criptografía RSA

Roberto Castillo 18546

Joonho Kim 18096

Matemática Discreta

Universidad del Valle de Guatemala

Facultad de Ciencias y Humanidades

Noviembre 2020

Implementación de RSA

Se hizo una implementación rudimentaria de RSA en Python. Esta no sigue al pie de la letra los estándares de seguridad establecidos para el algoritmo (RSA Laboratories, 2002). Pero sirve como ejercicio de técnicas de computación, como la exponenciación modular.

RSA se basa en el hecho de que, siendo los exponentes **e** y **d**, y el módulo **n** previamente calculados con cuidado,

$$(m^e (mod\ n))^d (mod\ n) \equiv m$$

Siendo **m** el mensaje a encriptar.

Esto significa que para comunicarse el emisor necesita entregar:

$$m^e (mod\ n)$$

Al receptor, que debe conocer **n** y **d** para poder computar:

$${m_{encriptado}}^d (mod\ n)$$

Y obtener el mensaje original.

Sin embargo, los exponentes **e** y **d** deben ser grandes para que el algoritmo sea seguro. Esto representa un reto computacional, ya que es muy complicado trabajar con números así de grandes, inclusive para computadoras. Por eso se utiliza el algoritmo de exponenciación modular, que produce el mismo resultado y es mucho más factible de computar (Khan Academy, 2020).

Finalmente, se necesitaba una manera de representar mensajes de texto como números para poder ser encriptados por RSA. En (RSA Laboratories, 2002), se propone un algoritmo conocido como un <<padding scheme>>. Este aumenta la seguridad de RSA, pero es complicado de implementar. Por eso se decidió no utilizarlo. Para reemplazarlo simplemente se obtuvo m del valor ASCII de cada letra +5 y, luego de que se obtiene m de la función de desencriptación, se le resta 5 para obtener el valor ASCII original. Esto asegura que RSA funcione para todos los valores de ASCII.

Referencias

Khan Academy. (2020, 11 20). *Modular exponentiation (article) | Khan Academy*. Retrieved from Khan Academy | Free Online Courses, Lessons & Practice: https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-exponentiation

RSA Laboratories. (2002). *PKCS #1 v2.1: RSA Cryptography Standard.* Bedford: RSA Laboratories.