Universidad Nacional Autónoma de México

Facultad de Ciencias

Criptografía y Seguridad

Práctica 7: Curvas elípticas y protocolo de Diffie-Hellman

Fecha de entrega: 08/12/2023

Equipo:

Criptonianos

Acosta Arzate Rubén - 317205776

Bernal Marquez Erick - 317042522

Deloya Andrade Ana Valeria - 317277582

Marco Antonio Rivera Silva - 318183583

1. Introducción

Las curvas elípticas han emergido como un pilar fundamental en el ámbito de la criptografía y seguridad. Aunque su conceptualización se remonta a siglos pasados, su aplicación en el mundo digital ha revolucionado la forma en que protegemos la información sensible y aseguramos las comunicaciones. Algunos criptografos aseguran que la criptografía de llave privada ha sido la única gran revolución sobre la seguridad, sin embargo cabe aclarar que este tipo de criptografía no ha desplazado a la de llave simétrica, ambos tipos son altamente utilizados hoy en día.

Estas curvas pueden describirse mediante ecuaciones específicas de tercer grado, aunque la forma más común es

$$y^2 = x^3 + px + q$$

por motivos de simplicidad y por propiedades específicas en la propia geometría de la curva.

2. Desarrollo

El codigo fue desarrollado utilizando Python, no se requiere instalar alguna depencia o biblioteca adicional. En general, para realizar el desarrollo de nuestro código bastó con seguir los métodos y algoritmos descritos en clase y en el *pdf*

El avance en las dos primeras clases, ***Punto*** y ***CurvaEliptica***, resultó ser bastante sencillo, ya que simplemente necesitamos seguir las definiciones que se discutieron en clase. Además, se requirió la implementación de una función para calcular el inverso modular, necesaria para llevar a cabo la suma de dos puntos, aunque este procedimiento también resultó ser relativamente simple.

Existen ciertos métodos en la clase ***CurvaElíptica*** que no son estrictamente necesarios para cifrar o descifrar, como por ejemplo, *tiene(p)*, *orden(p)* o *cofactor(p)*. A pesar de no ser esenciales para las operaciones criptográficas, estos métodos pueden ser bastante útiles para entender mejor el funcionamiento de las curvas.

La clase ***Entidad*** requiere de más atención y comprensión. Inicialmente, puede resultar algo confusa debido a la cantidad de llaves que recibe y genera a partir de otras. Esto, combinado con el protocolo de Diffie-Hellman, puede añadir cierta complejidad a la implementación. No es difícil; más bien, puede causar confusión si no se tiene un conocimiento fuerte del protocolo de intercambio de llaves. Sin embargo, al leer con atención y verificar que los algoritmos de generación de llaves funcionen correctamente se puede realizar el cifrado y descifrado de manera efectiva.

La implementación de los métodos de cifrado y descifrado conlleva también una atención particular, no es debido a las fórmulas utilizadas para estas operaciones ya que son realmente sencillas de implementar, sino por la forma en que python gestiona los diccionarios. En nuestra solución, empleamos un diccionario para asignar letras a puntos en la curva elíptica. Sin embargo, la tarea de asignar puntos a letras no es tan sencilla, ya que los objetos de tipo punto no proporcionan un *hash* que pueda usarse como clave en un diccionario. Una manera de solucionar este inconveniente implicaba la inclusión de los métodos *__hash__* y *__eq__* en la clase ***Punto***. No obstante, esta solución generaba un error al definir el punto al infinito como 'O', ya que carece de coordenadas x e y en el sentido convencional.

Para resolver este problema optamos por construir manualmente el diccionario que asigna puntos a letras, o más precisamente, tuplas a letras. Utilizamos tuplas porque estas pueden emplearse eficientemente como llaves en un diccionario. Además, nos permite realizar una conversión sencilla entre puntos y tuplas, y viceversa, al considerar únicamente la coordenada x como el primer elemento de la tupla y la coordenada y como el segundo elemento.

Las funciones *cifrar_aux* y *descifrar_aux* reciben un único elemento para realizar las operaciones de cifrado y descifrado, respectivamente. Posteriormente, los métodos 'cifrar' y 'descifrar' llaman a estas funciones para cada carácter presente en una cadena. En otras palabras, los métodos auxiliares para cifrar o descifrar procesan únicamente un elemento del mensaje, mientras que los métodos 'cifrar' o 'descifrar' lo hacen para la cadena completa.

3. Preguntas

- Cuando sumamos un punto P, con $-P$, que nos da como resultado el punto infinito, ¿Qué quiere decir que un punto sea infinito?

  Un punto infinito es un punto que se utiliza como frontera del conjunto de los números reales en el plano cartesiano y se asocia con una "línea en el infinito", que no es otra cosa que la línea vertical que nunca corta la curva elíptica pero para los algoritmos de cifrado y descifrado utilizando la curva simplemente funciona como nuestro elemento neutro para la suma de puntos de la curva elíptica.

- Cuando un punto P, al sumarlo consigo mismo, ¿qué característica tiene la recta que conecta a estos 2 puntos?

  Si queremos sumar P y Q en la curva entonces los unimos con una línea recta que generalmente corta con otro tercer punto en la misma curva, pero si P y Q son iguales entones será una tangente a la curva en ese punto.
  Por lo cual en algunas ocasiones (en especial cuando $y_1 = y_2 = 0$) hacer $P + P$ nos da como resultado el punto al infinito.

- ¿En qué otro cifrado se utiliza el protocolo Diffie-Hellman?

  Su aplicación más común es en combinación con otros esquemas criptográficos para establecer claves secretas compartidas en entornos donde la comunicación puede ser interceptada por terceros.

  Algunos de los cifrados donde se utiliza este protocolo son:

  - *Cifrado de Capa de Transporte (TLS/SSL)*
    Implementado en muchos protocolos de seguridad de capa de transporte, como TLS (Transport Layer Security) y su predecesor SSL (Secure Sockets Layer), para establecer

claves de sesión seguras entre un cliente y un servidor.

- *IPsec (Protocolo de Seguridad de Internet)*

  IPsec usa Diffie-Hellman para establecer claves compartidas en la fase de negociación de la Asociación de Seguridad (SA) durante la configuración de conexiones seguras de red.

- *SSH (Secure Shell)*

  De manera similar a TLS/SSL, se utiliza en SSH para establecer claves de sesión entre el cliente y el servidor para cifrar la comunicación.

- *PGP (Pretty Good Privacy)*

  En algunos casos puede utilizarse dentro del marco de PGP para el intercambio de claves.

- ¿Cuántos bits de una llave de una curva elíptica necesitamos para poder igualar la seguridad de una llave de RSA?

  La comparación directa entre el tamaño de las claves en cifrado de curva elíptica (ECC) y el cifrado RSA no es una tarea fácil ya que la seguridad de estos sistemas se basa en problemas matemáticos diferentes; logaritmo discreto para ECC y factorización de números primos para RSA. Sin embargo es posible hacer una estimación considerando la dificultad de los problemas *per se*, y no tanto en los bits.

  Por lo cual en términos de longitud de clave, una clave ECC de n bits se ***considera*** comparable en seguridad a una clave RSA de m bits de la siguiente manera.

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1: NIST Recommended Key Sizes

Figura 1: Comparación de bits entre ECC y RSA de acuerdo a NIST

- ¿Qué factor hace que una misma letra sea cifrada a dos diferentes puntos y que siga siendo recuperable?

  Una misma letra (o un mismo mensaje) puede cifrarse a dos diferentes puntos en la curva elíptica debido al proceso de generación de llaves y al uso del cifrado de curva elíptica.

  En el cifrado de curva elíptica, el punto resultante de la multiplicación de un punto base G por un escalar k (que es la clave privada) se utiliza como clave pública. Debido a la propiedad distributiva de la multiplicación en la curva elíptica, si tienes dos claves privadas diferentes k_1 y k_2, las correspondientes claves públicas $k_1 \times G$ y $k_2 \times G$ serán diferentes, incluso si cifran el mismo mensaje.

  El factor que permite recuperar el mensaje es la distributividad y conmutatividad de la propia aritmética de la curva y el intercambio de llaves entre entidades.

  Recordemos que nuestro mensaje cifrado se representa como $\{kG, P_m + kP_b\}$, donde kG es k veces el punto G, P_m es el mensaje que queremos cifrar y kP_b es k veces la llave pública del destinatario B. Así, para recuperar el mensaje solo debemos restar n veces la llave secreta por kG, es decir $P_m + kP_b - n_b(kG)$, donde n_b es la llave secreta del destinatario.

La distributividad y conmutatividad nos permite hacer lo siguiente $P_m + kP_b - n_b(kG) = P_m + kn_bG - n_bkG = P_m$, ya que $P_b = n_bG$, pudiendo así recuperar el mensaje original.

4. Conclusiones

Las curvas elípticas han demostrado ser una herramienta poderosa en el ámbito de la criptografía. Su aplicación ofrece ventajas significativas en términos de eficiencia y seguridad en comparación con otros métodos criptográficos tradicionales (como RSA o de llave simétrica). La naturaleza de las propiedades matemáticas de las curvas elípticas tales como distributibidad y conmutatividad hacen de ellas una solución simple para aplicarse a la criptografía moderna.

Además de ser versátiles y adaptables en distintos campos matemáticos como teoría de grupos, teoría de números, análisis complejo, entre otras; el caso que nos interesa es la criptografía donde se utilizan en distintos ámbitos como las firmas digitales, cifrado en la capa de transporte, protocolos de internet, etc.

La implementación de la práctica fue relativamente sencilla, de hecho los problemas que se nos presentaron eran más por cuestiones del lenguaje que de los algoritmos y el proceso en sí mismo, el cual algunas veces resultaba confuso. Aunque todo estaba descrito detalladamente en la teoría, la traducción a código a veces requería una cuidadosa consideración de los detalles matemáticos y de las particularidades de los objetos en Python.

5. Referencias

- *Colaboradores de los proyectos Wikimedia. (2006, 18 de septiembre). Punto del infinito - Wikipedia, la enciclopedia libre. Wikipedia, la enciclopedia libre.*
  https://es.wikipedia.org/wiki/Punto_del_infinito

- *7.3.2.3 Intercambio de claves de Diffie-Hellman. (s.f.). Institut Sa Palomera – ESO, Batxillerat,*

Batxillerat nocturn i Cicles d'Informàtica a Blanes - INICI.
https://www.sapalomera.cat/moodlecf/RS/4/course/module7/7.3.2.3/7.3.2.3.html

- *Mendoza, W. (2017, 11 mayo). ¿Cómo se compara ECC con RSA?*
  https://www.certificadosdigitales.net/como-se-compara-ecc-con-rsa/

- *Mates Mike. (2022, 9 febrero). El problema del MILENIO sobre CURVAS ELÍPTICAS [Vídeo]. YouTube.* https://www.youtube.com/watch?v=9mR_h9ufs4E