Álgebra abstracta en la criptografía

Patarroyo Valbuena, Maria Camila

Hernández Quintana, Valentina

Rodriguez Salazar, Juan Sebastian

Matemáticas Aplicadas y Ciencias de la Computación

Universidad del Rosario

Álgebra abstracta y codificación

Prof. Mauro Artigiani

21 Marzo 2024

Índice

1. Presentación del grupo **2**

2. Acuerdos **2**

3. Descripción del proyecto **3**

4. Objetivos **3**

5. Cronograma **3**

6. Fundamentos Matemáticos **4**
6.1. Teoría de Grupos y Anillos 4
6.2. Aritmética Modular 5

7. Algotimos criptográficos **5**

8. Explicación detallada de los códigos y ejemplos **7**
8.1. RSA 7
8.1.1. Funciones 7
8.1.2. Ejemplo RSA 7
8.2. ElGamal 8
8.2.1. Funciones 8
8.3. Ejemplo ElGamal 10
8.3.1. Ejemplo Simple 10
8.3.2. Ejemplo ElGamal 10

9. Resultados **10**

10. Análisis de los Algoritmos **11**
10.1. Eficiencia 11
10.1.1. RSA: 11
10.1.2. ElGamal: 11
10.2. Seguridad 11
10.2.1. RSA 11
10.2.2. ElGamal 11
10.3. Aplicaciones Prácticas 12
10.3.1. RSA 12
10.3.2. ElGamal 12
10.4. Comparación 12

11. Conclusiones **12**

12. Recursos **13**

1. Presentación del grupo

I. Maria Camila Patarroyo Valbuena
Rol: Bibliotecario
Correo: mariac.patarroyo@urosario.edu.co

II. Juan Sebastian Bernal Rojas
Rol: Coordinador
Correo: juanseba.bernal@urosario.edu.co

III. Juan Sebastian Rodriguez Salazar
Rol: Ordaor
Correo: juansebasti.rodrig10@urosario.edu.co

IV. Valentina Hernández Quintana
Rol: Anotador
Correo: valentina.hernndezq@urosario.edu.co

2. Acuerdos

Acuerdos entre los participantes del grupo:

Reuniones:

Nos reuniremos semanalmente, ya sea en persona o en una plataforma virtual, para discutir avances y retroalimentaciones del proyecto. Se notificará al grupo con al menos 1 hora de anticipación si no podemos asistir.

Comunicación:

Utilizaremos WhatsApp como canal principal de comunicación y nos comprometemos a responder a los mensajes del grupo en un plazo máximo de 1 día.

Solución de conflictos:

Abordaremos los conflictos de manera abierta y respetuosa, buscando escuchar todas las perspectivas. En caso de necesidad, designaremos a Sebastián Bernal como mediador neutral.

Responsabilidades y contribuciones:

Cada miembro del grupo tendrá responsabilidades específicas, detalladas en un documento compartido junto con un cronograma para cada fase del proyecto. Cumpliremos con los plazos acordados y comunicaremos cualquier retraso.

Contingencias:

Ante la incapacidad temporal de un miembro para cumplir con sus responsabilidades, redistribuiremos equitativamente las tareas. Si alguien desaparece o incumple repetidamente, informaremos al grupo y tomaremos medidas para garantizar la continuidad del proyecto.

Compromiso:

Reconocemos la importancia de nuestro compromiso individual con el éxito del proyecto. Trabajaremos colaborativamente, apoyándonos mutuamente y manteniendo una cultura de responsabilidad y respeto.

3. Descripción del proyecto

El manejo de la información en la era digital es crucial, especialmente al considerar su transmisión a través de canales inseguros como Internet o redes locales. En tales entornos, la información está expuesta a posibles interceptaciones por parte de terceros no autorizados, lo que provoca riesgos significativos de violación de la confidencialidad y la integridad de los datos.

Para abordar estos desafíos, es esencial adherirse a los principios fundamentales de seguridad de la información. La confidencialidad garantiza que solo las partes autorizadas puedan acceder a la información, mientras que la integridad asegura que los datos permanezcan intactos durante su transmisión. Por otro lado, la autenticidad garantiza la veracidad de la fuente del mensaje, lo que puede lograrse mediante el uso de firmas digitales.

En este contexto, la criptografía surge como un pilar en la seguridad de la información. Esta disciplina no solo protege los archivos de datos, sino también los medios de transporte, al permitir la ocultación del contenido de un mensaje mediante técnicas de cifrado. Dentro de estas técnicas, el cifrado asimétrico desempeña un papel crucial al utilizar dos claves distintas para cifrar y descifrar datos.

4. Objetivos

I. Estudiar, entender y explicar el uso de la aritmetica modular y algebra abstracta en los algoritmos de RSA, Elgamal y Diffie-Hellman.

II. Desarrollar e implementar algoritmos de RSA y ELgamal, para asi luego compararlos y deducir cual es el mejor en un sistema criptografico.

5. Cronograma

- Semana 3-4: Preparación Inicial:
  - Identificación de participantes del grupo y asignación de roles.

- Acuerdo sobre horarios de reunión y comunicación.
- Investigación inicial sobre criptografía y algoritmos RSA y ElGamal para escribir la descripción del proyecto y establecer los objetivos.

- Semana 5-6: Investigación y Planificación:
  -Revisión de literatura relacionada con criptografía y algoritmos criptográficos.
  - Desarrollo de un plan detallado para la implementación de RSA y ElGamal.
  -Selección de herramientas de software y recursos necesarios.

- Semana 7-9: Implementación de Algoritmos:
  - Desarrollo del código para el algoritmo RSA.
  - Desarrollo del código para el algoritmo ElGamal

- Semana 10-12: Continuación del documento:
  - Dejar en el dcumento una explicación detallada de los fundamntos matemáticos de RSA Y ElGamal.
  - Realizar un análisis de los algoritmos y compararlos.

- Semana 14-15: Planeación de la entrega final:
  - Planear las pruebas de los algoritmos y explicar el paso a paso de como se hara.
  - Entregar un preliminar del documento.

- Semana 16-17: Correcciones y pruebas:
  - Realizar las correcciones pertinentes del preliminar.
  - Llevar a cabo las pruebas de los algoritmos.

- Semana 18: Documentación de resultados:
  - Documentar los resultados de las pruebas.
  - Preparar entrga final y sustntación.

6. Fundamentos Matemáticos

La comprensión de los fundamentos matemáticos es esencial para abordar los algoritmos criptográficos RSA y ElGamal. Estos algoritmos se basan en conceptos clave de la teoría de grupos y anillos, que proporcionan el marco matemático para su implementación y análisis.

6.1. Teoría de Grupos y Anillos

Grupos: Un grupo es un conjunto G junto con una operación binaria $\cdot$ que cumple con las siguientes propiedades: cerradura, existencia de un elemento neutro, existencia de inversos y asociatividad. En criptografía, los grupos abelianos (donde la operación es conmutativa) son especialmente importantes.

Anillos: Un anillo es un conjunto R equipado con dos operaciones binarias (adición y multiplicación) que cumplen con propiedades como asociatividad, existencia de un elemento neutro para la adición y distribución de la multiplicación sobre la adición. En criptografía, trabajamos frecuentemente con anillos conmutativos, donde la multiplicación también es conmutativa.

6.2. Aritmética Modular

La aritmética modular es un sistema de aritmética para enteros, donde los números "vuelven.ªl principio después de alcanzar un cierto valor, el módulo. La operación modular más común es el residuo de la división de dos números. Si a y b son enteros y n es un entero positivo, entonces $a \equiv b \pmod{n}$ significa que a y b dejan el mismo residuo al dividirse por n

7. Algotimos criptográficos

A continuación se mostrara cómo cada parte de los algoritmos RSA y ElGamal se relaciona con la teoría de grupos y anillos, y cómo las operaciones modulares y las propiedades de estos conceptos garantizan la seguridad de la criptografía. La selección adecuada de números primos y la aplicación de operaciones algebraicas adecuadas garantizan la robustez de estos algoritmos frente a ataques criptoanalíticos. Sin embargo en este proyecto no se abordara a profundidad la selección adecuada de números primos.

RSA: En RSA, la aritmética modular desempeña un papel fundamental. Este algoritmo se basa en el grupo multiplicativo de residuos módulo n, donde n es el producto de dos números primos grandes p y q. Aquí, la selección adecuada de los primos p y q garantiza la seguridad del sistema. Además, el conjunto de residuos módulo n forma un anillo conmutativo con la operación de multiplicación, lo que permite la generación y manipulación de claves.

I. **Generación de Claves:**

- Primero, elegimos dos números primos grandes, digamos p y q. Estos números deben ser lo suficientemente grandes para asegurar que n, que es el producto de p y q, sea difícil de factorizar.
- Luego, calculamos $n = p \times q$. Este número n será parte de la clave pública y se utiliza en el proceso de cifrado.
- Después, calculamos $\phi(n)$, que es $(p-1) \times (q-1)$. Este valor es importante porque nos ayudará a elegir un número especial llamado exponente público.
- Elegimos un número e como exponente público, que debe ser coprimo con $\phi(n)$. Esto significa que e y $\phi(n)$ no tienen divisores comunes aparte del 1.
- Finalmente, calculamos el exponente privado d, que es el inverso multiplicativo de e módulo $\phi(n)$. Esto significa que d es el número que, cuando se multiplica por e y luego se toma el módulo $\phi(n)$, da 1.

II. **Cifrado y Descifrado:**

- Para cifrar un mensaje m, lo convertimos primero a un número.
- Luego, elevamos ese número a la potencia de e (el exponente público) y tomamos el residuo de la división por n. El resultado es el mensaje cifrado c.

- Para descifrar, tomamos el mensaje cifrado c y lo elevamos a la potencia de d (el exponente privado).
- Luego, tomamos el residuo de la división por n. El resultado es el mensaje original m.

ElGamal En álgebra abstracta, un grupo cíclico es un conjunto de elementos que pueden ser generados por un solo elemento, llamado generador. En el caso del algoritmo de cifrado ElGamal, se utiliza un grupo cíclico de orden primo, específicamente el conjunto de los enteros módulo un número primo p, denotado como $\mathbb{Z}_p^*$. Las operaciones de cifrado y descifrado en ElGamal se realizan en este grupo cíclico. ElGamal se basa en la aritmética modular dentro de un grupo abeliano bajo la multiplicación módulo p. La operación de exponenciación modular permite la generación de claves y el cifrado de mensajes. Un aspecto fundamental de la seguridad de ElGamal es la dificultad del problema del logaritmo discreto, que consiste en encontrar el exponente a tal que $g^a = h$, dado un generador g y un elemento h del grupo. Además, el Pequeño Teorema de Fermat, que establece que si p es un número primo y a es un número entero que no es divisible por p, entonces a elevado a la $(p-1)$ es congruente a 1 módulo p (es decir, $a^{p-1} \equiv 1 \mod p$), juega un papel crucial en la teoría subyacente al algoritmo. En resumen, ElGamal utiliza estas propiedades algebraicas para proporcionar un marco seguro para el cifrado y descifrado de mensajes.

I. **Generación de Claves:**

- Elegimos un número primo grande p. Este número será el módulo para nuestras operaciones.
- Luego, seleccionamos un número α que actúa como nuestro generador en el grupo multiplicativo de residuos módulo p.
- Elegimos un número privado d, que debe ser un número aleatorio entre 1 y $p-1$.
- Calculamos el exponente público β como α elevado a la potencia de d y luego tomamos el residuo módulo p. Este β será parte de nuestra clave pública.

II. **Cifrado y Descifrado:**

- Para cifrar un mensaje m, seleccionamos un número aleatorio k entre 1 y $p-1$.
- Calculamos dos valores: $KE = \alpha^k \mod p$ y $C_2 = m \cdot \beta^k \mod p$. Aquí, KE y C_2 son las partes del mensaje cifrado.
- Para descifrar el mensaje, tomamos los valores cifrados KE y C_2.
- Calculamos $KE^d \mod p$ para obtener un valor intermedio.
- Multiplicamos C_2 por el inverso de este valor intermedio para recuperar el mensaje original m.

8. Explicación detallada de los códigos y ejemplos

8.1. RSA

8.1.1. Funciones

El código implementa el algoritmo RSA en Python. Mostraremos la explicación de las funciones y el flujo del programa:

- `divide_blocks(message, block_size)`: Esta función divide un mensaje en bloques de tamaño específico. Se utiliza para dividir un mensaje en bloques antes de cifrarlo.
- `letter_ascii(message)` y `ascii_letter(message_ascii)`: Estas funciones convierten un mensaje de texto en su representación ASCII y viceversa. Esto es útil para convertir el mensaje en un formato que se puede cifrar y luego volver a convertirlo al formato original después de descifrarlo.
- `maximo_comun_divisor(a, b)`: Calcula el máximo común divisor (MCD) de dos números utilizando el algoritmo de Euclides. Esta función se utiliza para calcular la clave privada d en la generación de claves RSA.
- `find_mod_inv(a, m)`: Encuentra el inverso multiplicativo de a en módulo m. Esta función se utiliza para calcular la clave privada d en la generación de claves RSA.
- `genLlaves(p, q)`: Esta función genera un par de claves pública y privada utilizando dos números primos p y q. Primero calcula el producto n de p y q, luego calcula la función totient de Euler $\phi(n)$, elige un exponente público e y calcula el exponente privado d.
- `rsa_encrypt(x, kpub)` y `rsa_decrypt(y, kpriv)`: Estas funciones cifran y descifran un bloque de datos utilizando claves pública y privada RSA, respectivamente.
- `main()`: La función principal del programa. En esta función se generan las claves pública y privada, se cifra y descifra un mensaje de ejemplo, y se imprime la salida.

8.1.2. Ejemplo RSA

I. **Generación de Claves:** En este paso, se generan las claves pública y privada. La clave pública consiste en un par de números (e, n), mientras que la clave privada consiste en un par de números (d, n). El valor de n se calcula como el producto de dos números primos grandes distintos, p y q. El valor de $\phi(n)$, conocido como la función totient de Euler, se calcula como $(p-1)(q-1)$.

   Supongamos que elegimos $p = 17$ y $q = 19$. Entonces, $n = p \times q = 17 \times 19 = 323$. Además, $\phi(n) = (p-1)(q-1) = 16 \times 18 = 288$.

Para seleccionar el exponente de cifrado e, se elige un número entero positivo e que sea coprimo con $\phi(n)$, es decir, e y $\phi(n)$ no tienen factores comunes excepto 1. Un valor comúnmente utilizado para e es 65537 debido a su eficiencia en operaciones de exponenciación.

Si tomamos $e = 17$, entonces e y $\phi(n)$ son coprimos ya que su máximo común divisor (MCD) es 1.

El exponente de descifrado d se calcula como el inverso multiplicativo de e en módulo $\phi(n)$. Es decir, d es el número tal que $e \cdot d \equiv 1 \mod \phi(n)$.

Para encontrar d, podemos utilizar el algoritmo de Euclides extendido o buscar directamente el inverso multiplicativo de e en módulo $\phi(n)$. Si $e = 17$, entonces $d = 233$ ya que $17 \cdot 233 \equiv 1 \mod 288$.

II. **Cifrado:** El cifrado RSA se realiza elevando el mensaje M a la potencia e y luego tomando el residuo módulo n. El mensaje cifrado se denota como C.

Supongamos que queremos cifrar el mensaje $M = 123$. Entonces, el mensaje cifrado C se calcula como $C \equiv M^e \mod n$.

$$C \equiv 123^{17} \mod 323$$

$$C \equiv 308 \mod 323$$

Por lo tanto, el mensaje cifrado C es 308.

III. **Descifrado:** El descifrado RSA se realiza elevando el mensaje cifrado C a la potencia d y luego tomando el residuo módulo n. El mensaje original M se recupera.

Para descifrar el mensaje cifrado $C = 308$, calculamos $M \equiv C^d \mod n$.

$$M \equiv 308^{233} \mod 323$$

$$M \equiv 123 \mod 323$$

Por lo tanto, el mensaje original M es 123.

8.2. ElGamal

8.2.1. Funciones

- **Función** `maximo_comun_divisor`:
  Esta función implementa el algoritmo de Euclides para encontrar el máximo común divisor (MCD) de dos números enteros a y b. Repetidamente reemplaza a con b y b con el residuo de a dividido por b hasta que b sea 0. El valor final de a es el MCD.

- **Función** `find_mod_inv`:
  Esta función encuentra el inverso modular de a bajo el módulo m. Recorre los valores de x desde 1 hasta $m-1$ y verifica si $(a \cdot x)\,\%m == 1$. Si encuentra tal x, lo retorna como el inverso modular. Si no lo encuentra, lanza una excepción.

- **Función** `genLlaves`:
  Esta función genera las llaves públicas y privadas para RSA.

  - Calcula n como el producto de dos números primos p y q.
  - Calcula ϕ (función de Euler) como $(p-1) \cdot (q-1)$.
  - Elige un número e al azar tal que $2 \leq e < \phi$ y que sea coprimo con ϕ (g = 1).
  - Calcula el inverso modular d de e bajo ϕ.
  - Retorna la llave pública (e, n) y la llave privada (d, n).

- **Función** `rsa_encrypt`:
  Esta función cifra un mensaje x usando la llave pública *kpub*. Usa la función `pow` para calcular $x^e\,\%n$.

- **Función** `rsa_decrypt`:
  Esta función descifra un mensaje y usando la llave privada *kpriv*. Usa la función `pow` para calcular $y^d\,\%n$.

- **Función** `read_pdf_file`:
  Esta función lee el contenido de un archivo PDF y retorna su texto.

- **Función** `letter_ascii`:
  Esta función convierte un mensaje en una lista de sus valores ASCII, rellenando cada valor con ceros a la izquierda para que cada valor tenga exactamente tres dígitos.

- **Función** `ascii_letter`:
  Esta función convierte una lista de valores ASCII (en formato de cadena de tres dígitos) de vuelta a un mensaje de texto.

- **Función** `main`:
  Esta función coordina todas las anteriores para realizar el proceso completo de cifrado y descifrado de un mensaje extraído de un archivo PDF utilizando el algoritmo RSA.

  - Genera las llaves públicas y privadas RSA.
  - Lee el contenido de un archivo PDF.
  - Convierte el mensaje a su representación ASCII.
  - Cifra los bloques ASCII.
  - Descifra los bloques cifrados.
  - Convierte los bloques descifrados de vuelta al mensaje original.
  - Mide y muestra el tiempo de cifrado y descifrado.

8.3. Ejemplo ElGamal

8.3.1. Ejemplo Simple

8.3.2. Ejemplo ElGamal

Ejemplo Simple
Alicia elige los valores:

- $p = 17$ (primo aleatorio y suponemos que $p-1 = 16$ tiene un factor primo grande lo cual no es cierto)
- $g = 3$ (generador aleatorio)
- $a = 6$ (clave privada aleatoria)

Alicia calcula

$$K = g^a \mod p = 3^6 \mod 17 = 15$$

(valor calculado)

Por tanto la clave pública será $(g, p, K) = (3, 17, 15)$.

Bruce tiene el texto claro $m = 9$ (el cual está entre 1 y $p-1$). Bruce escoge un $b = 5$ aleatorio entre 2 y $p-1$. Bruce calcula

- $y_1 = g^b \mod p = 3^5 \mod 17 = 5$
- $y_2 = K^b m \mod p = 15^5 \cdot 9 \mod 17 = 1$

Por lo que el texto cifrado es $C_b(m, b) = (y_1 = 5, y_2 = 1)$.

Para descifrar, Alicia usa la clave privada ($a = 6$) y el Pequeño Teorema de Fermat:

$$m = y_1^{p-1-a} y_2 \mod p = 5^{10} \cdot 1 \mod 17 = 9$$

9. Resultados

Cifrado y almacenamiento del mensaje:

- Una vez que las claves públicas se han generado y almacenado, el computador puede cifrar un mensaje utilizando su propia clave pública.
- El mensaje cifrado se guarda en un archivo en el mismo computador.

Lectura y descifrado del mensaje:

- El computador lee el mensaje cifrado desde el archivo.
- Utiliza su clave privada correspondiente para descifrar el mensaje.
- Una vez descifrado, el mensaje original está disponible para su visualización.

10. Análisis de los Algoritmos

En esta sección, se realizará un análisis comparativo de los algoritmos RSA y ElGamal, centrándonos en su funcionamiento, eficiencia, seguridad y aplicaciones prácticas.

10.1. Eficiencia

10.1.1. RSA:

- La eficiencia del RSA está dominada por las operaciones de exponenciación modular, las cuales pueden ser optimizadas mediante métodos como la exponenciación rápida.
- La generación de claves es especialmente costosa debido a la necesidad de encontrar números primos grandes y realizar la factorización.

10.1.2. ElGamal:

- Similar al RSA, la eficiencia de ElGamal está dominada por las operaciones de exponenciación modular.
- ElGamal puede ser más eficiente en términos de tamaño de clave debido a que no requiere números primos tan grandes como RSA.

10.2. Seguridad

10.2.1. RSA

- La seguridad del RSA se basa en la dificultad de factorizar grandes números compuestos. Si n es suficientemente grande (típicamente al menos 2048 bits), la factorización es computacionalmente inviable con la tecnología actual.
- La seguridad también depende de la elección adecuada de los números primos p y q y de mantener la clave privada en secreto.

10.2.2. ElGamal

- La seguridad de ElGamal se basa en la dificultad del problema del logaritmo discreto en un grupo finito. Si los parámetros son elegidos correctamente (p.ej., un número primo p de al menos 2048 bits), es resistente a ataques criptoanalíticos conocidos.
- La elección de p y g y la seguridad del canal de transmisión del número aleatorio k son cruciales.

10.3. Aplicaciones Prácticas

10.3.1. RSA

- RSA es ampliamente utilizado en la seguridad de la comunicación, incluyendo cifrado de datos, firmas digitales y autenticación de usuarios. Ejemplos incluyen HTTPS, SSL/TLS, y protocolos de correo electrónico seguro como S/MIME.

10.3.2. ElGamal

- ElGamal se utiliza en protocolos de intercambio de claves como el protocolo de Diffie-Hellman, y en esquemas de firma digital, como el Digital Signature Algorithm (DSA).

10.4. Comparación

Ambos algoritmos tienen fortalezas y debilidades en términos de eficiencia y seguridad. La elección entre RSA y ElGamal depende de las necesidades específicas de seguridad y eficiencia de una aplicación particular.

- **Eficiencia:** ElGamal puede ser más eficiente que RSA en términos de tamaño de clave, pero ambos requieren operaciones de exponenciación modular.
- **Seguridad:** RSA se basa en la factorización de números grandes, mientras que ElGamal se basa en el logaritmo discreto. Ambos son seguros si se eligen correctamente los parámetros y se mantienen las claves privadas seguras.
- **Aplicaciones:** RSA es muy utilizado en aplicaciones de cifrado y firmas digitales, mientras que ElGamal es popular en protocolos de intercambio de claves y firmas digitales.

11. Conclusiones

RSA, un gigante en el campo de la criptografía, ha demostrado su robustez y confiabilidad a lo largo de los años. Su implementación sencilla y su amplia compatibilidad con diversos protocolos de seguridad, como SSL/TLS y PGP, lo convierten en una opción atractiva para muchas aplicaciones. Además, con longitudes de clave adecuadas, RSA ofrece un nivel de seguridad sólido y bien establecido.

Por otro lado, ElGamal también presenta ventajas, especialmente en términos de eficiencia. Sin embargo, su menor adopción y soporte pueden representar desafíos a la hora de integrarlo en sistemas existentes y en aplicaciones críticas.

Al considerar los tiempos de ejecución, hemos observado que RSA demuestra un rendimiento ligeramente más lento tanto en el cifrado como en el descifrado en comparación con ElGamal. Aunque estos tiempos son importantes, no son el único factor a tener en cuenta al tomar una decisión.

- RSA Encryption Time: 0.049701 segundos
- ElGamal Encryption Time: 0.00751280784606933 segundos

En última instancia, hemos decidido optar por RSA como nuestro principal algoritmo criptográfico. Su combinación de seguridad sólida, facilidad de implementación y amplio soporte lo convierten en la elección más adecuada para nuestros propósitos. Aunque ElGamal puede ofrecer ciertas ventajas en términos de eficiencia, consideramos que la fiabilidad y la versatilidad de RSA son más valiosas para nuestras necesidades de seguridad informática.

12. Recursos

i. Gaurav Mittal, Sunil Kumar, Shiv Narain, y Sandeep Kumar. (2020). GROUP RING BASED PUBLIC KEY CRYPTOSYSTEMS.

ii. Stanoyevitch, A. (2013). Introduction to Cryptography with Mathematical Foundations and Computer Implementations. Routledge.

iii. Kurose, J., Ross, K. Computer Networking: a top-down approach. 7th ed. 2016.

iv. Gayoso Martínez, Victor. Hernandez Encinas, Luis. Martín Muñoz, Agustín. Criptografia con Curvas Elipticas. 2018.

v. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2), 120-126.

vi. ElGamal, T. (1985). A Public Key Cryptosystem and A Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4), 469-472.

vii. Koblitz, N. (1987). A Course in Number Theory and Cryptography. *Springer-Verlag.*

viii. Kaliski, B. (1993). A Survey of Encryption Standards. *IEEE Micro*, 13(6), 74-81.

ix. Boneh, D. (1999). Twenty Years of Attacks on the RSA Cryptosystem. *Notices of the AMS*, 46(2), 203-213.

x. Schneier, B. (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C. *John Wiley & Sons.*

xi. Stinson, D. R. (2006). Cryptography: Theory and Practice. *Chapman & Hall/CRC.*

xii. Katz, J., & Lindell, Y. (2014). Introduction to Modern Cryptography. *Chapman & Hall/CRC*