Algoritmo ElGammal

Ramírez León Christian Yael
Instituto Politécnico Nacional - ESFM, U.P.A.L.M, San Pedro Zacatenco, 07738, CDMX, México

1. Introducción

En la era digital, la seguridad de la información es un aspecto fundamental para garantizar la privacidad y la confiabilidad de las comunicaciones. La criptografía juega un papel clave en este contexto, permitiendo proteger datos sensibles frente a accesos no autorizados. Una de las herramientas más importantes dentro de la criptografía moderna es el uso de algoritmos asimétricos, que emplean un par de claves distintas: una clave pública para cifrar los mensajes y una clave privada para descifrarlos.

El algoritmo de ElGamal, desarrollado por Taher ElGamal en 1985, es uno de los pilares de la criptografía asimétrica. Este algoritmo se basa en la dificultad computacional del problema del logaritmo discreto, una operación matemática que es fácil de realizar en una dirección (exponenciación modular) pero extremadamente compleja de invertir sin conocer información adicional.

ElGamal no solo se utiliza para cifrar mensajes, sino también para generar firmas digitales seguras, lo que lo convierte en una herramienta versátil en diversos sistemas de seguridad. Además, sus variantes y principios han sido integrados en protocolos ampliamente adoptados, como el sistema de cifrado GNU Privacy Guard (GPG) y otros esquemas criptográficos.

2. Marco Teórico

Un sistema criptográfico es un sistema que consiste en un algoritmo de cifrado y uno de descifrado y tres espacios de textos bien definidos, texto plano, texto cifrado y texto clave[3]. Normalmente dado un texto clave, el algoritmo de cifrado mapeará un texto plano a un texto cifrado que usualmente es único, esto a partir de una clave, por lo que la clave es esencial, para definir un paso particular del cifrado.

Los algoritmos de cifrado se clasifican principalmente en tres: cifrado simétrico, asimétrico y funciones hash criptográficas. De manera general a partir de un cifrado simétrico se transmitirá el mensaje, mientras que en el cifrado asimétrico se transmitirán claves para el algoritmo simétrico, esto debido a que los algoritmos para el primer tipo suelen ser más veloces, mientras que los segundos son más seguros en la transmisión.

2.1. Principios de Kerckhoff

Los principios de Kerckhoff son las propiedades deseables que tenga un sistema criptográfico, estos son:

1. Si el sistema no es teóricamente irrompible, al menos debe serlo en la práctica.
2. La efectividad del sistema no debe depender de que su diseño permanezca en secreto.
3. La clave debe ser fácilmente memorizable de manera que no haya que recurrir a notas escritas.
4. Los criptogramas deberán dar resultados alfanuméricos.
5. El sistema debe ser operable por una única persona.
6. El sistema debe ser fácil de utilizar.

2.2. Cifrado simétrico

Es un método el cual utiliza la misma clave para cifrar y descifrar un mensaje.

Dados los conjuntos $\mathcal{K}$ el cual será el espacio para las posibles claves, $\mathcal{M}$ para los posibles mensajes y $\mathcal{C}$ para los posibles textos cifrados. El cifrado está dado por la siguiente función:

$$e : \mathcal{K} \times \mathcal{M} \to \mathcal{C} \tag{1}$$

Donde el dominio $\mathcal{K} \times \mathcal{M}$ es el conjunto de los pares ordenados (k, m), donde k es la clave del cifrado y m el mensaje que se desea transmitir. Para descifrar el mensaje se tiene la siguiente función:

$$d : \mathcal{K} \times \mathcal{C} \to \mathcal{M} \tag{2}$$

Dichas funciones cumplen la característica que:

$$d(k, e(k, m)) = m \quad \forall k \in \mathcal{K}, m \in \mathcal{M}$$

Es conveniente marcar la dependencia de las funciones con respecto a la clave k:

$$e_k : \mathcal{M} \to \mathcal{C} \quad d_k : \mathcal{C} \to \mathcal{M}$$

Por lo que podemos notar que de esta forma d_k es la función inversa de e_k, ya que satisface:

$$d_k(e_k(m)) \to m \quad \forall m \in \mathcal{M}$$

Además se la función e_k es una función inyectiva, ya que, si se tiene $m, m' \in \mathcal{M}$, tales que $e_k(m) = e_k(m')$, entonces:

$$m = d_k(e_k(m)) = d_k(e_k(m')) = m'$$

2.3. Cifrado asimétrico

Del mismo modo que en el cifrado simétrico se requieren espacios $\mathcal{K}, \mathcal{M}, \mathcal{C}$, pero en está ocasión la clave va a estar dada como un par ordenado, es decir:

$$k = (k_{priv}, k_{pub}) \tag{3}$$

De modo que k_{priv} se le llama clave privada y k_{pub} es clave pública.
Para cada clave pública hay un función de cifrado correspondiente:

$$e_{k_{pub}} : \mathcal{M} \to \mathcal{C} \tag{4}$$

Adicionalmente para la llave pública:

$$d_{k_{priv}} : \mathcal{C} \to \mathcal{M} \tag{5}$$

Dichas funciones cumplen la condición:

$$d_{k_{priv}}(e_{k_{pub}}(m)) = m \quad \forall m \in \mathcal{M} \tag{6}$$

Se le dice a la llave privada puerta trampa o trampilla *(trapdoor)* porque proporciona una forma fácil de computar la inversa de $e_{k_{pub}}$, esto se muestra en la figura 1.

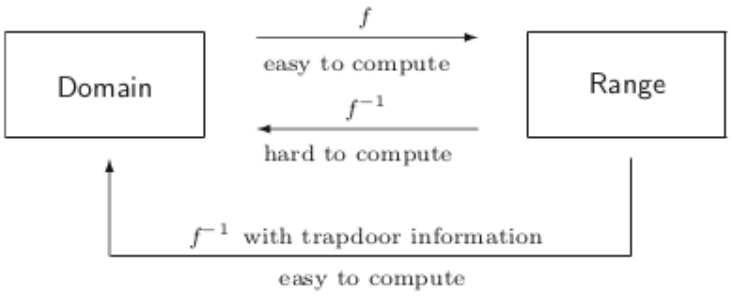

Figura 1: Ilustración de una función con trampilla.

Este cifrado permite enviar un mensaje a partir de un canal inseguro, gracias a la clave privada.

2.4. Aritmética modular

Sea p un número primo. El conjunto de enteros $\mathbb{Z}_p$, está formado por los restos de la división de enteros entre p. La operación de suma y producto módulo p convierte a $\mathbb{Z}_p$ en un campo finito. Es decir, cada elemento distinto de cero en $\mathbb{F}_p$ tiene un inverso multiplicativo. El subconjunto de elementos distintos de cero, denotado por $\mathbb{Z}_p^*$, forma un grupo bajo la multiplicación módulo p, este grupo tiene exactamente $p-1$ elementos, y por el teorema pequeño de Fermat, cualquier elemento $a \in \mathbb{F}_p$ satisface:

$$a^{p-1} = 1 \tag{7}$$

2.4.1. Elemento primitivo y ciclos

Un elemento $g \in \mathbb{Z}_p$ se llama **primitivo** si sus potencias generan todos los elementos de $\mathbb{Z}_p$. En otras palabras, g es primitivo si:

$$1, g, g^2, \ldots, g^{p-2} \in \mathbb{Z}_p$$

Forman la lista completa de los elementos en $\mathbb{Z}_p^*$.

2.4.2. Algoritmo rápido para la potenciación

Este algoritmo es utilizado para calcular $g^a \pmod m$ con $g \in \mathbb{Z}_m$ y $a \in \mathbb{N}$. Se realiza a partir de los siguientes pasos:

1. Realizar la expansión binaria de a, es decir:

$$a = a_0 + 2^1 a_1 + \cdots + 2^r a_r \quad \text{con } a_0, a_1, \ldots, a_r \in 0, 1$$

2. Computar las potencias de $g^{2^i} \pmod m$, para $i = 0, 1, \ldots, r$, por medio de elevar al cuadrado:

$$\begin{aligned}
b_0 &\equiv g \pmod m \\
b_1 &\equiv b_0^2 \equiv g^2 \pmod m \\
b_2 &\equiv b_1^2 \equiv (g^2)^2 \pmod m \\
b_3 &\equiv b_2^2 \equiv (g^2)^3 \pmod m \\
&\vdots \\
b_r &\equiv b_{r-1}^2 \equiv (g^2)^{r-1} \pmod m
\end{aligned}$$

3. Computar g^a de la siguiente forma:

$$\begin{aligned}
g^a &= g^{a_0 + 2^1 a_1 + \cdots + 2^r a_r} \\
&= \prod_{i=0}^{r} (g^{2^i})^{a_i} \\
&\equiv \prod_{i=0}^{r} b_i^{a_i} \pmod m
\end{aligned}$$

2.5. El problema del logaritmo discreto

Dado un número primo p, un elemento primitivo g en $\mathbb{Z}_p$, y un elemento $h \in \mathbb{Z}_p$, el problema del logaritmo discreto consiste en encontrar el entero x tal que:

$$g^x \equiv h \pmod p \tag{8}$$

2.6. Intercambio de clave Diffie-Hellman

El protocolo Diffie-Hellman es un método criptográfico para el intercambio de claves que permite a dos partes establecer una clave compartida sobre un canal inseguro. Este protocolo se basa en la dificultad computacional del problema del logaritmo discreto.

Sea p un número primo y g un generador (elemento primitivo) de $\mathbb{Z}_p^*$. Los pasos del protocolo son los siguientes:

1. Los valores p y g se acuerdan y se hacen públicos.
2. Se elijen las claves privadas:
   - La parte A elige una clave privada a, donde $1 \leq a \leq p-2$.
   - La parte B elige una clave privada b, donde $1 \leq b \leq p-2$.
3. Se calculan las claves publicas:
   - A calcula $A_{\text{pub}} \equiv g^a \pmod{p}$ y envía A_{pub} a B.
   - B calcula $B_{\text{pub}} \equiv g^b \pmod{p}$ y envía B_{pub} a A.
4. Se calcula una clave compartida
   - A calcula $K \equiv (B_{\text{pub}})^a \pmod{p}$.
   - B calcula $K \equiv (A_{\text{pub}})^b \pmod{p}$.

3. Sistema criptográfico ElGammal

El algoritmo de cifrado de clave pública ElGammal está basado en el problema del logaritmo discreto y está altamente relacionado con el intercambio de clave Diffie-Hoffman[1], mencionado en la sección 2.6.

3.1. Explicación del algoritmo

Supongamos que dos personas Alicia y Bob quieren intercambiar un mensaje a través de un canal inseguro, el cual está siendo interceptado por un enemigo Eve, Alicia escogerá dos números p un primo suficientemente grande, y $g \in \mathbb{Z}_p$ elemento primitivo, ambos números serán públicos, posteriormente escogerá un número $a \in [[1, p-2]]$ la cuál funcionará como clave privada, para calcular:

$$A \equiv g^a \pmod{p} \tag{9}$$

Dicho número calculado será la clave pública del sistema.

Ahora supongamos que el mensaje de Bob, es un entero $m \in [[2, p-2]]$ (puede ser un texto que se haya codificado por medio de ASCII) y además escogerá un número aleatorio $k \pmod{p}$ que será utilizado para cifrar el mensaje. Usando la clave pública obtenida en la ecuación 9, computará dos cantidades:

$$c_1 \equiv g^k \pmod{p} \quad \wedge \quad c_2 \equiv mA^k \pmod{p} \tag{10}$$

De modo que Bob enviará el par ordenado (c_1, c_2) a Alicia. Alicia lo descifrará de la siguiente manera:

$$x \equiv (c^a)^{-1} \pmod{p} \tag{11}$$

Esto a partir del algoritmo 2.4.2, calculando $c^{p-1-a} \pmod{p}$, ya que

$$(c^a)^{-1} \equiv c^{p-1-a} \pmod{p}$$

Posteriormente realizar el producto:

$$x \cdot c_2 \equiv m \pmod{p} \tag{12}$$

Podemos notar que en efecto es el mensaje ya que:

$$\begin{aligned} x \cdot c_2 &\equiv c_1^{-1} \cdot c_2 \\ &\equiv (g^{ak})^{-1} \cdot (mA^k) \\ &\equiv (g^{ak})^{-1} \cdot (mg^{ak}) \\ &\equiv m \end{aligned}$$

3.2. Implementación del algoritmo en C.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

int Modulo(int a, int p);

int FastPot(int potencia, int base, int p);

int ClavePublica(int clave_privada, int g, int p);

void cifradoGammal(int *c1, int *c2, int clave_pub, int g, int p, int m);
int DescifradoGammal(int *c1, int *c2, int clave_privada, int p);

int charToInt(char c);
char IntToChar(int n);

int main() {
  int *c1 = (int *)malloc(sizeof(int));
  int *c2 = (int *)malloc(sizeof(int));
  int g = 5, p = 503;
  int clave_privada = rand() % (p - 3) + 2;
  int clave_pub = ClavePublica(clave_privada, g, p);
  int i;
```

```
25    int m;
26
27    char *mensaje, *mensaje_descifrado;
28    int *mensaje_c1, *mensaje_c2;
29
30    mensaje = (char *)malloc(100 * sizeof(char));
31    printf("Ingrese algo a cifrar: ");
32    scanf("%*c%[^\n]", mensaje);
33
34    int mensaje_len = strlen(mensaje);
35    mensaje_c1 = (int *)malloc(mensaje_len * sizeof(int));
36    mensaje_c2 = (int *)malloc(mensaje_len * sizeof(int));
37    mensaje_descifrado = (char *)malloc((mensaje_len + 1) * sizeof(char));
38
39    srand(time(NULL));
40
41    // Cifrado
42    for (i = 0; i < mensaje_len; i++) {
43      m = charToInt(mensaje[i]);
44      cifradoGammal(c1, c2, clave_pub, g, p, m);
45      mensaje_c1[i] = *c1;
46      mensaje_c2[i] = *c2;
47    }
48
49    // Descifrado
50    for (i = 0; i < mensaje_len; i++) {
51      *c1 = mensaje_c1[i];
52      *c2 = mensaje_c2[i];
53      m = DescifradoGammal(c1, c2, clave_privada, p);
54      mensaje_descifrado[i] = IntToChar(m);
55    }
56
57    mensaje_descifrado[mensaje_len] = '\0';
58
59    printf("Mensaje original: %s\n", mensaje);
60    printf("Mensaje descifrado: %s\n", mensaje_descifrado);
61
62    free(c1);
63    free(c2);
64    free(mensaje);
65    free(mensaje_c1);
66    free(mensaje_c2);
67    free(mensaje_descifrado);
68
69    return 0;
70  }
71
72  int Modulo(int a, int p) {
73    return ((a % p) + p) % p;
74  }
75
76  int FastPot(int potencia, int base, int p) {
77    int resultado = 1, b;
78    b = Modulo(base, p);
```

```
  resultado = potencia % 2 ? b : 1;
  potencia = potencia / 2;

  while (potencia != 0) {
    b = Modulo(b * b, p);
    if (potencia % 2) {
      resultado = Modulo(resultado * b, p);
    }
    potencia = potencia / 2;
  }
  return resultado;
}

int ClavePublica(int clave_privada, int g, int p) {
  return FastPot(clave_privada, g, p);
}

void cifradoGammal(int *c1, int *c2, int clave_pub, int g, int p, int m) {
  int k = (rand() % (p - 3)) + 2; // Generar k entre 2 y p-2
  *c1 = FastPot(k, g, p);
  *c2 = Modulo(m * FastPot(k, clave_pub, p), p);
}

int DescifradoGammal(int *c1, int *c2, int clave_privada, int p) {
  int x = FastPot(p - 1 - clave_privada, *c1, p);
  return Modulo(x * (*c2), p);
}

int charToInt(char c) {
  return (int)c;
}

char IntToChar(int n) {
  return (char)n;
}
```

Referencias

[1] Hoffstein, Jeffrey & Phipher, Jill & Silverman, Joseph H., *An Introduction to Mathematical Criptography*, segunda edición, Springer, 2014.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*, septima edición, Pearson, 2016.

[3] Henk C. A. Van Tilborg & Sushil Jajodia, *Encyclopedia of Cryptography and Security*, segunda edición, Springer, 2011.

[4] IBM, "Encryption", disponible en: `https://www.ibm.com/mx-es/topics/encryption`

[5] Cryptography Academy, "ElGamal," [En línea]. Disponible en: `https://cryptographyacademy.com/elgamal/`. [Accedido: 18-dic-2024].