Intercambio de claves de Diffie y Hellman

Adrián Racero Serrano
Juan Manuel Cardeñosa Borrego

Índice

1 Introducción

El algoritmo Diffie-Hellman debe su nombre a sus creadores Whitfield Diffie y Martin Hellman. Creado en 1976, es uno de los protocolos de intercambio de claves más antiguos que todavía se siguen usando en la actualidad. Sus creadores fueron galardonados con el premio A.M. Turing 2015 por este trabajo, con el que revolucionaron por completo la seguridad informática.

Este algoritmo permite a dos usuarios cualesquiera intercambiar, de forma confidencial, una clave secreta K (o de sesión) para posteriormente cifrar de forma simétrica los mensajes entre ellos dos.

2 Algoritmo

El funcionamiento de este algoritmo es más sencillo de lo que parece y se usa frecuentemente en protocolos y aplicaciones de encriptado de datos, como SSL (Secure Sockets Layer), SSH (Secure Shell) o VPN (Virtual Private Network). Este algoritmo permite que dos entidades (A y B) puedan generar una clave KAB de forma simultánea, y sin enviarla por el canal de comunicaciones.

1) Para ello, A y B necesitan establecer y compartir valores comunes, como un valor q primo y una raíz primitiva α de q.

Para todo primo q existe un elemento $a \in (\mathbb{Z}/q\mathbb{Z})^\times$ con $\text{ord}(a) = q - 1 = \phi(q) = \#(\mathbb{Z}/q\mathbb{Z})^\times$. En otras palabras,

$$(\mathbb{Z}/q\mathbb{Z})^\times = \{1, a, a^2, \ldots, a^{q-2}\}.$$

Este a se llama una **raíz primitiva** módulo q. Ejemplo:

$$q = 3,\ \alpha = 2;\ 2^1 \ mod\ 3 = 1,\ 2^2 \ mod\ 3 = 2$$

2) Tanto A como B generan sus claves privadas ($X_{A/B}$) y públicas ($Y_{A/B}$) teniendo en cuenta que: $Y_A \equiv \alpha^{XA} \pmod q$, donde $0 \leq X_A \leq (q-1)$, X_A es el logaritmo discreto de Y_A, y se representa $dlog_{\alpha,q}\ (Y_A)$. Por tanto, la efectividad del algoritmo depende de la dificultad de computar logaritmos discretos.

3) Tanto A como B comparten sus respectivas claves públicas (Y_A/Y_B).

4) Tanto A como B generan de forma "mágica" la clave de sesión teniendo en cuenta: KAB = $(Y_B)^{XA}$ mod q. Ya tengo ssh

3 Código

Definition **class Diffie_Hellman** means ...

```
from sympy import isprime
import random

def generar_clave_privada(q, a):
    # Generar una clave privada aleatoria en el rango [2, q-2]
    clave_privada = random.randint(2, q - 2)
    return clave_privada

def generar_clave_publica(q, a, clave_privada):
    # Calcular la clave publica usando la formula: clave_publica = (a^clave_privada) % q
    clave_publica = pow(a, clave_privada, q)
    return clave_publica

def generar_clave_compartida(q, clave_publica_otro, clave_privada_propia):
    # Calcular la clave compartida usando la formula: clave_compartida = (clave_publica_otro^clave_privada_propia) % q
    clave_compartida = pow(clave_publica_otro, clave_privada_propia, q)
    return clave_compartida

# Parametros compartidos publicamente (q y a)
q = 23
a = 5

# Generar claves privadas y publicas para ambas partes
clave_privada_alice = generar_clave_privada(q, a)
clave_publica_alice = generar_clave_publica(q, a, clave_privada_alice)

clave_privada_bob = generar_clave_privada(q, a)
clave_publica_bob = generar_clave_publica(q, ax, clave_privada_bob)

# Intercambio de claves publicas
clave_compartida_alice = generar_clave_compartida(q, clave_publica_bob, clave_privada_alice)
clave_compartida_bob = generar_clave_compartida(q, clave_publica_alice, clave_privada_bob)

```

```
34 # Ambas partes deberian tener la misma clave compartida
35 print("Clave compartida por Alice:", clave_compartida_alice)
36 print("Clave compartida por Bob:", clave_compartida_bob)
37
38 # Verificar que ambas partes tienen la misma clave compartida
39 assert clave_compartida_alice == clave_compartida_bob
```

4 Conclusión

a