Laboratorio 7: Diffie Hellman

Javiera Fuentes, Nicolás Vergara, Pablo Gajardo, Camilo Muñoz y Vicente Moya

November 11, 2024

1 Introducción

1.1 Contexto

El laboratorio 7 corresponde a la segunda parte del curso de *Criptografía y Seguridad en la Información (TEL-252)*, en el cual se exploran los fundamentos del protocolo Diffie-Hellman, uno de los algoritmos clásicos para el intercambio seguro de claves. Este laboratorio se enfoca en el estudio del logaritmo discreto y su relevancia en la seguridad del protocolo, además de la implementación y análisis de funciones para el cálculo de claves compartidas.

1.2 Objetivos

Los objetivos de este laboratorio son:

- Comprender el funcionamiento y la seguridad del protocolo Diffie-Hellman para el intercambio de claves.
- Implementar en SageMath funciones que permitan simular el intercambio de claves Diffie-Hellman.
- Estudiar la vulnerabilidad del protocolo frente a ataques basados en el cálculo del logaritmo discreto.
- Generar y utilizar campos finitos en SageMath, aplicándolos en la implementación de claves compartidas.
- Validar la implementación mediante ejercicios prácticos que simulen un intercambio de claves seguro.

Todo el código desarrollado para éste laboratorio puede encontrarse en el siguiente repositorio de github.

2 Metodología

2.1 Flujo del laboratorio

- Se inicia el laboratorio con una introducción al protocolo Diffie Hellman, explicando el intercambio de claves seguras y las propiedades matemáticas de los números primos, con ejemplos prácticos presentados por el profesor y el ayudante.
- Se procede a implementar los ejercicios usando Sage, calculando valores secretos y simulando el intercambio de claves con parámetros específicos. Se explora y comprende la importancia del orden multiplicativo y la elección de parámetros seguros.
- El trabajo consiste en generar parámetros y calcular claves compartidas. Se discuten las implicaciones de los logaritmos discretos y cómo podrían vulnerar el protocolo si los campos finitos no son lo suficientemente grandes.
- Al finalizar, se resumen los resultados y se presentan las soluciones desarrolladas, enviando los archivos completos para la fecha de entrega.

2.2 Ejercicios ejecutados

Para éste laboratorio no se desarrollaron ejercicios durante la clase.

3 Ejercicios prácticos

3.1 Pregunta 1

a) Supón que eres Bob y has acordado con Alice los parámetros de dominio $p = 70849$ y $g = 2$. Supón además que Alice ha enviado el valor $X = 39674$. Calcula un valor secreto y y computa Y, y el secreto compartido.

```
    def modulo_p(num, g, p):

    conv = (g^num)%p

    return conv

    p = 70849
    g = 2

    X = 39674 #Clave publica de Alice

    valor_y = 24221 #Valor secreto de Bob

    Y = modulo_p(valor_y, g, p) #Clave publica de Bob. Se envia a Alice

    K = (X^(valor_y))%p

    print("Pregunta A")
    print("Valor 'y' calculado: " + str(valor_y))
    print("Valor 'Y' para enviar a Alice: " + str(Y))
    print("Clave compartida 'K': " + str(K))

    print("\n")

    /////output/////
    '''
    Valor 'y' calculado: 24221
    Valor 'Y' para enviar a Alice: 48033
    Clave compartida 'K': 65654
    '''
```

b) Supón que Alice y Bob han acordado los parámetros de dominio $p = 6779$ y $g = 3$. Además, supón que Alice elige el valor secreto $x = 384$ y Bob elige el valor secreto $y = 152$. Realiza un intercambio de claves simulado como en el ejemplo.

```
    p = 6779
    g = 3

    x = 384
    y = 152

    X = (g^x)%p
```

```
    Y = (g^y)%p

    print("Pregunta B")
    print("Valor 'X' generado por Bob para Alice: " + str(X))
    print("Valor 'Y' generado por Alice para Bob: " + str(Y))
    print("Es la clave 'K' igual para ambos?: " + str( (X^y)%p == (Y^x)%p) )
    print("Clave 'K' generada por Alice y Bob: " + str((X^y)%p) + " y " + str((Y^x)%p) +
    ↪  " respectivamente")

    print("\n")

    /////output/////
    '''
    Valor 'X' generado por Bob para Alice: 4437
    Valor 'Y' generado por Alice para Bob: 6267
    Es la clave 'K' igual para ambos?: True
    Clave 'K' generada por Alice y Bob: 6397 y 6397 respectivamente
    '''
```

c) Encuentra un primo q y un primo p tal que $p = 2q + 1$, encuentra un elemento en el campo finito con p elementos que tenga orden multiplicativa q. Realiza un intercambio de claves DH simulado como en los ejemplos.

```
def primos_q_y_p():
    for i in range(100):
        lista = list(prime_range((i+1)*10, (i+1)*10 + 100))

        for k in range(len(lista)):

            q = lista[k]
            p = 2*q + 1

            if (is_prime(p)):

                return q, p
    return -1

def generar_g(q, p):

    g = 2

    while g<p:

        if ((g^q)%p == 1):

            for i in range(q):

                if((g^(i+1))%p == 1 and i+1 != q):

                    break

                if(i+1 == q):
                    return g
```

```
            g = g + 1

        return 0


q, p = primos_q_y_p()


g = generar_g(q, p)

x = 384
y = 152

X = (g^x)%p
Y = (g^y)%p
print("Pregunta C")
print("Valor 'X' generado por Bob para Alice: " + str(X))
print("Valor 'Y' generado por Alice para Bob: " + str(Y))
print("Es la clave 'K' igual para ambos?: " + str( (X^y)%p == (Y^x)%p) )
print("Clave 'K' generada por Alice y Bob: " + str((X^y)%p) + " y " + str((Y^x)%p) +
↪  " respectivamente")

/////output/////
'''
Valor 'X' generado por Bob para Alice: 12
Valor 'Y' generado por Alice para Bob: 6
Es la clave 'K' igual para ambos?: True
Clave 'K' generada por Alice y Bob: 4 y 4 respectivamente
'''
```

3.2 Pregunta 2

a) Implementa una función en Sage que tome un límite y devuelva 4 elementos: p, q, g y F. Donde p y q son primos, tal que $p = 2q + 1$, g es un entero con orden multiplicativa q en el campo finito con p elementos, y F es un objeto de campo finito con p elementos.

```
 from random import randint

def primos_q_y_p():
    for i in range(100):
        lista = list(prime_range((i+1)*10, (i+1)*10 + 100))

        for k in range(len(lista)):

            q = lista[k]
            p = 2*q + 1

            if (is_prime(p)):

                return q, p
    return -1
```

```
16
17      def Cuatro_Elementos(limite):
18
19          g = 2
20
21          for i in range(100):
22              lista = list(prime_range(5, limite))
23
24              for k in range(len(lista)):
25
26                  q = lista[k]
27                  p = 2*q + 1
28
29                  if (is_prime(p)):
30
31                      break
32
33          F = GF(p)
34
35          while g<p:
36
37              if ((g^q)%p == 1):
38
39                  for i in range(q):
40
41                      if((g^(i+1))%p == 1 and i+1 != q):
42
43                          break
44
45                      if(i+1 == q):
46                          return g, p, q, F
47
48              g = g + 1
49
50          return 0
51
52
53      g, p, q, F = Cuatro_Elementos(100)
54
55      /////output/////
56      '''
57      Los elementos entregados son:
58      (3, 11, 5, Finite Field of size 11)
59      '''
```

b) Implementa una función en Sage que tome la salida de la función de la parte (a) y devuelva el par (X, x) donde $X = g^x \mod p$ y x es mayor que 1 y menor que q.

```
1       def Par_Xx_Yy(g, p, q, F):
2
3           x = randint(2, q)
4           X = F(g)^x
5
```

```
        return x,X

    x, X = Par_Xx_Yy(g,p,q,F)
    print("\n")
    print("Pregunta B")
    print(f"La variable x es igual a: {x}")
    print(f"La variable X es igual a: {X}")

    /////output/////
    '''
    La variable x es igual a: 4
    La variable X es igual a: 4
    '''
```

c) Implementa una función en Sage que tome un valor público de la otra parte en el intercambio de claves DH y el valor secreto, y devuelva el secreto compartido.

```
    def Clave_Secreta(x,Y,p):

        K = (Y^x)%p

        return K

    print("\n")
    print("Pregunta C")
    print("La clave generada con una clave compartida igual a " + str(39674) + " es: " +
    ↪  str(Clave_Secreta(x, 39674,p)))

    /////output/////
    '''
    La clave generada con una clave compartida igual a 39674 es: 4
    '''
```

d) Muestra un ejemplo de intercambio de claves usando tus funciones de las partes (a)–(c).

```
    print("Se utiliza 'x' e 'y' al azar ")

    g, p, q, F = Cuatro_Elementos(100)

    print(f"g, p y q será igual a: {g}, {p}, y {q} respectivamente")

    x, X = Par_Xx_Yy(g,p,q,F)
    y, Y = Par_Xx_Yy(g,p,q,F)

    print(f"Entonces, los pares son: x={x}, X={X}, y={y} e Y={Y}")

    K_Bob = Clave_Secreta(x,Y,p)
    K_Alice = Clave_Secreta(y,X,p)

    print(f"Entonces, las claves generadas por ambos son: Alice->{K_Alice} y
    ↪  Bob->{K_Bob}")

    /////output/////
    '''
```

```
    Se utiliza 'x' e 'y' al azar
    g, p y q será igual a: 3, 11, y 5 respectivamente
    Entonces, los pares son: x=4, X=4, y=3 e Y=5
    Entonces, las claves generadas pro ambos son: Alice->9 y Bob->9
    '''
```

3.3 Pregunta 3

El propósito de esta pregunta es usar Sage para explorar cómo resolver el logaritmo discreto puede romper el protocolo DH.
En Sage, si a es un elemento de un campo finito y g lo genera, entonces si el orden del campo finito es lo suficientemente pequeño, $a.\log(g)$ devolverá el logaritmo discreto de g con respecto a a.
Utiliza esta funcionalidad para resolver los siguientes problemas:

a) Supón que $p = 499$, $g = 7$, y $X = 297$. Encuentra x tal que $X = g^x$.

```
p = 499
g = 7
X = 297

F = GF(p)
x = F(X).log(F(g))

print("Pregunta A")
print(f"El valor x obtenido por logaritmo discreto es: " + str(x))

/////output/////
'''
Pregunta A
El valor x obtenido por logaritmo discreto es: 362
'''
```

b) Supón que $p = 863$, $g = 5$, $X = 543$, y $Y = 239$. Encuentra x y y tal que $X = g^x$ y $Y = g^y$.

```
p = 863
g = 5
X = 543
Y = 239

F = GF(p)

x = F(X).log(F(g))
y = F(Y).log(F(g))

print("\n")
print("Pregunta B")
print(f"Los valores secretos de cada sujeto son: 'x'={x} e 'y'={y}")

/////output/////
'''
Pregunta B
Los valores secretos de cada sujeto son: 'x'=536 e 'y'=762
'''
```

c) Supón que $p = 7589$, $g = 2$, $X = 6075$ y $Y = 1318$. Encuentra el valor secreto compartido.

```
p = 7589
g = 2
X = 6075
Y = 1318

F = GF(p)

x = F(X).log(F(g))
y = F(Y).log(F(g))

print("\n")
print("Pregunta C")
print("Para comprobar, usar ambos casos")

K1 = (X^y)%p
K2 = (Y^x)%p

print(f"K1 = {K1} y K2 = {K2}, lo que indica que esta correcto")

/////output/////
'''
Pregunta C
Para comprobar, usar ambos casos
K1 = 6803 y K2 = 6803, lo que indica que esta correcto
'''
```

4 Conclusiones

4.1 Resumen

El laboratorio permitió un estudio profundo del protocolo Diffie-Hellman y su implementación práctica en SageMath. A través de ejercicios prácticos, se realizaron simulaciones de intercambio de claves y se verificaron las propiedades matemáticas de este protocolo, incluyendo el uso de logaritmos discretos y campos finitos. La metodología aplicada permitió la implementación y validación de cada paso del intercambio de claves, confirmando la correcta aplicación de los conceptos criptográficos fundamentales.

4.2 Reflexión

Este laboratorio destacó la importancia de los parámetros seguros en la criptografía. Los resultados obtenidos evidenciaron la vulnerabilidad del protocolo Diffie-Hellman frente a ataques de logaritmo discreto cuando los campos finitos son de orden insuficiente. El uso de SageMath facilitó la exploración de estos aspectos, permitiendo identificar cómo un atacante podría deducir el valor secreto compartido en configuraciones inseguras. Este conocimiento refuerza la relevancia de elegir adecuadamente los parámetros en sistemas reales para garantizar la confidencialidad en el intercambio de claves.

References

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. London, UK: Pearson, 2017.