Texto, Carta

Descripción generada automáticamente**Icono

Descripción generada automáticamente con confianza mediaUn conjunto de letras negras en un fondo negro

Descripción generada automáticamente con confianza baja**

**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

**Ingeniería en Sistemas Computacionales**

**Prof. Nidia Asunción Cortes Duarte**

**Practica 5.  
(Diffie-Hellman con Curvas Elípticas)**

**Equipo:**

**Rodríguez Olmos Noé**

**Israel Sebastián Espinosa Sánchez**

**Esdras Josué Gutiérrez Magallanes**

**Grupo: 7CM1**

**Selected Topics in Cryptography**

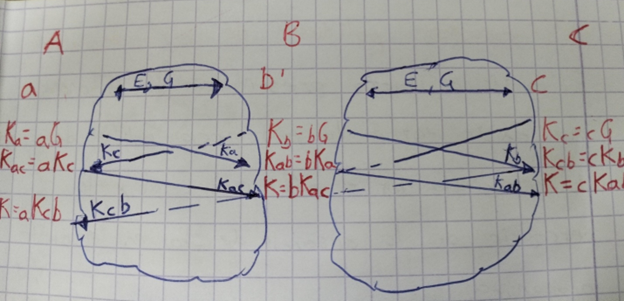
**14 de mayo del 2024**

**Práctica** **5**

**DESCRIPCION DEL ALGORITMO**

ECDH es un protocolo de acuerdo de claves que permite a dos partes, cada una con un par de claves pública-privada de curva elíptica, establecer un secreto compartido a través de un canal inseguro. Este secreto compartido se utiliza para derivar otra clave simétrica. El protocolo ECDH es una variante del protocolo Diffie-Hellman que utiliza criptografía de curva elíptica. ECDH deriva un valor secreto compartido a partir de una clave secreta poseída por una Entidad A y una clave pública poseída por una Entidad B, cuando las claves comparten los mismos parámetros de dominio de la curva elíptica. La Entidad A puede ser el iniciador de una transacción de acuerdo de claves, o el respondedor en un esquema. Si dos entidades realizan correctamente las mismas operaciones con claves correspondientes como entradas, se produce el mismo valor secreto compartido.

**DIAGRAMA ECDH**



**CARACTERISTICAS DE LOS DISPOSITIVOS**

|  |  |  |  |
| --- | --- | --- | --- |
| PC | RAM | S.O. | PROCESADOR |
| A (Noé) | **16 GB** | **W 11** | **RYZEN 7 (4800U)** |
| B (Esdras) | **4 GB** | **W 10** | **INTEL PENTIUM R** |
| C (Israel) | **8 GB** | **W 11** | **INTEL CORE I5 (10 GEN)** |

**LENGUAJE, BIBLIOTECA Y CURVA**

**Lenguaje:** PYTHON ver 3.11

**Biblioteca:** TYNYEC import EC

**Curva:** curve = registry.get\_curve('brainpoolP512r1')

**PROBLEMAS DURANTE EL DESARROLLO**

Inicialmente usábamos una librería de criptografía que no nos permitía generar los puntos y por ende las llaves no se generaban bien.

También que no nos permitía manejar los archivos por que al guardar se usaba utf-8 y eso manipulaba la información final al momento de leerla.

**PRUEBAS DE IMPLEMENTACION**

Para solucionar el problema de la curva, cambiamos de librería, en vez de usar *cryptodome* empezamos a usar *tynyec* que permite manipular los puntos de la curva de manera mas eficaz al realizar las operaciones.

**CODIGO**

from tkinter import \*

from tkinter import filedialog

from tkinter import messagebox

from tinyec import registry

import secrets

from tinyec import ec

global PrivKey1

global PubKey1

global curve

def compress(pubKey):

    x\_hex = hex(pubKey.x)[2:]

    y\_hex = hex(pubKey.y)[2:]

    compressed = x\_hex + y\_hex

    return compressed

def decompress(compressedPubKey):

    x = int(compressedPubKey[:128], 16)

    y = int(compressedPubKey[128:], 16)

    return x, y

def FirstKey(): #Generates public and private keys

    global PrivKey1

    global PubKey1

    global curve

    curve = registry.get\_curve('brainpoolP512r1')

    # Generation of private and public key of the first user

    PrivKey1 = secrets.randbelow(curve.field.n)

    PubKey1 = PrivKey1 \* curve.g

    # Ensure the public key is compressed correctly

    PubKey1\_compressed = compress(PubKey1)

    while len(PubKey1\_compressed) != 256:

        PrivKey1 = secrets.randbelow(curve.field.n)

        PubKey1 = PrivKey1 \* curve.g

        PubKey1\_compressed = compress(PubKey1)

    # File name

    file\_name = "Llave\_publica.txt"

    # Write the compressed variable to the file

    with open(file\_name, "w") as file:

        file.write(PubKey1\_compressed)

        file.close()

    messagebox.showinfo("Success", "Public key saved successfully")

def SecondKey(): #Generates the second key to send

    global curve

    global PrivKey1

    compressedPubKey = str(Kc.get())

    x, y = decompress(compressedPubKey)

    public2 = ec.Point(curve, x, y)

    # Calculation of the shared key 1

    first\_SharedKey = PrivKey1 \* public2

    KeyShared1 = compress(first\_SharedKey)

    # File name

    file\_name = "Llave\_compartida.txt"

    # Write the compressed variable to the file

    with open(file\_name, "w") as file:

        file.write(KeyShared1)

    messagebox.showinfo("Success", "Shared key saved successfully")

def ThirdKey(): #Generates the final key for Diffie-Hellman

    global curve

    global PrivKey1

    compressedPubKey = str(Kcb.get())

    x, y = decompress(compressedPubKey)

    public3 = ec.Point(curve, x, y)

    # Calculation of the shared key 1

    first\_SharedKey = PrivKey1 \* public3

    KeyShared2 = compress(first\_SharedKey)

    # File name

    file\_name = "LLaveCompartidaFinal.txt"

    # Write the compressed variable to the file

    with open(file\_name, "w") as file:

        file.write(KeyShared2)

# Tkinter window

window = Tk()

window.geometry("350x300")

window.title("P5")

button1 = Button(window, text="Calculate public key", command=FirstKey)

button1.grid(row=4, column=1, ipadx=10, ipady=5)

Kc = StringVar()

entry1 = Entry(window, textvariable=Kc, width=50)

entry1.grid(row=6, column=1)

text1 = Label(window, text="K1:")

text1.grid(row=6, column=0, padx=5, pady=1)

button2 = Button(window, text="Calculate K2", command=SecondKey)

button2.grid(row=7, column=1, ipadx=10, ipady=5)

Kcb = StringVar()

entry2 = Entry(window, textvariable=Kcb, width=50)

entry2.grid(row=9, column=1)

text2 = Label(window, text="K2:")

text2.grid(row=9, column=0, padx=5, pady=1)

button3 = Button(window, text="Generate shared key", command=ThirdKey)

button3.grid(row=10, column=1, ipadx=10, ipady=5)

window.mainloop()

**CAPTURAS**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteINTERFAZ**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamentePrueba 1**

Llave publica A: 1b80dc5875b60aa042fea9ce26c2499fd2122cb901ef15c5e545f59b9d3c311dd68e01c5902e92ef40c96f6a7519e9c17401ffb49c6ca016dcc0b7da5ed963fd6bf738ffc17d32c3cfdd46b2c228f3465f2bef459eeab98113d259f6497514d804a276ced9a4fc6ad56b415510313d8576d61e33bc9dc081d57faa3a24f99cb9

Llave publica B:

9aea7583e8566eb1b6d690f84953b7c7c62b7f841cb8c8591913aacfebd59e995300a1b3e24b22b9e59d7f8cd2301a2633cf4c0a1475eda99467d19b6bde7cf9254669bd4eebedfadd3451a1e28b05a7fb2e19da4c948cc43e682cfb8bfcd7e099b09615199273d02ab73a04ecb7f9db02ca47f5a3f114e87cd4344fb0ab4af4

Llave publica C:

9817ef9cd1bdf45d7e34ca7284b64115c5fdbb87978201cbed9ddaf47bb627e977e155194ba7464ae1e98646a8dc749d21d59b8b4e6dfd0471831d177e56c917928c644e36382f635426e901befb0182a81ae30400397fe5c0b49cb9aa7f78c0763002548821d14b0bf3144164896c702f36c58406f6791e251a85aacea8e20f

Llave AB:

6fbed505c14dbe4426b3423057ffc76cfa0048a0ea697bbdebb0c5c4a75ad6ea52de4484905ba6f12668e43d566ca941e601894ebf2f89397962f364adc16cb09982840c6d3df3dc900fa5b762d326192ffe8b5bbfaabb2f88489307be1bc979011bfdb6c16f75c24d3129c34df96b4cf0132b861c95f80e9de70d665ac6e690

Llave BC:

95741d7aba7766e8217320923db626a05259e9a6b66076caad459050c54c26d13ddbb723765383ac8a023cc638b550ac700100891065e450174ea8acdbff0ee95047721f19dc9c2c4f196b42395df5c5a848326b291b9e861c9c4ee88d0a0ca94284a7353b3776c64ce10ee49dbe530b4a330bfd1695b651484d48df45c59a0e

Llave CA:

2dcef5f96aa50785ca32be9836dda6076f180d26d74be1bc4302c0d6e589406ff5e5edb9cc49fd030265836cf40e14c0e3dee61db908c5f7f74bb431bd4f79c2a1ae3fc5ac9c26c41e1c0c929dd82a3b67ab7a644e8c316c21744c98c535e61bceb83325079d869f84daebe991381e5cbfba7c68e3de295e5f3ba3953f4ef7e6

Secreto compartido:

b07a741128f0ea8c884bd78aff99a471edf32cf82974215db56dacb012422ff4d4a042bdc2173f0aaafd4aef3cd18d63cf0f3597cb45ee8ade381fc6c542cfc11f5d94188a9f43078f59d0396c5df29156f59863556478d39dffa9daf5112604cbcbe057e32c176becfe9d5f4565a623c11f9b6afaeb1f937edfbd055bfeb70

**Prueba 2**

Llave A:

557d8f8c00456b47d9db3494b2effea542c96ea3ec5d9467e0f7a31307064b9c4f518e4251b7b781fda01576682111406da4c159b150cdeea9f9fac2d97f94f76aa580e4aa6ab4946ddae6d92578c853101ca5e7261831ab768b3aa356d79576f0cbc46790d6e89ac83a829f4696324658790a0623f2a92740ffc9a1a5ebfd94

Llave B:

13f798fad7779459efd94f6e2c7c4c29f8cab928249d9383cbbf14e4c41172e910c16979c4c92a55d6f5553d260a94a81850fe62a8e503eb9af51e643b9fad847f52fd0113b8f877dd96ab478442f26d9e6a01a2af124c582d60b85f684b1f0beddbcff4e3fa9f3df9b27787468ed33514400fb11320ff6bba37fb3e06c610ba

Llave C:

737bbf31946cc3235c8a94b8795538afe29f425a3d881b179c0b5df7dd181edd03f9a1dc1850376b240d7d86afec446854dab6ca43ca85df48b1e020aa2bf75a4b346d6c71104d3a5e072d554a874c4b7463fa360a9b4127f167afcd635c36ae95fc97ef232173907b28070f189a5954e904ea79a539f711e8c6b9e6f5df0d9b

Llave AB:

212ee08ea812aa15fd3a5a7ea5a968929e4ddfa2667923574e9abc04bcff9b52a476f1ed95ad61ce97dbb48b441f8fe50626338d18a396d82487d0b87244a7c3a144b4eaa0d34dc37a9f401446b3de614018b8c920f05eac6ff65fa2cce4d6aa19a65d72853db24ba66b2e240213b28efda7a5ed290cf3c44b2e183118ca63a1

Llave BC:

404165af186c55b1952ce9db7a9f1899729063fa0c9519db87da016e58b9ea599993eff30b7aa7538ed49baf919e2bc676421812a7a18ec9e8fd489584e6c56c8c82d902c28de375c704bcc886c08a66c659beead3231e3b7200162688a8ba77f240be0ad803ebcef6ad4beca4a0f8140f871a628d1bc80eaa89a53e307f89ea

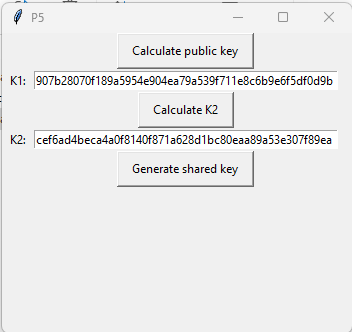
Llave CA:

29f9f6339bf9bf35a5e89f6c52072857a87381813d67a5cf206d9b00628e9b5890dae201ad36628b34d0723c98e97f64c1ccc5fe52d1ad1a7598b007b34e0fd49e63b279dd59c13ec0a25d54af7de5f6a927d9fea610ee1ed99957ecf04ea94700349a73b10071416672e4c0ad51985de29ae6d08b7b2e67e823c7bac89c2564

Secreto Compartido:

89c822b06ccbe0f4345ee08f5144633a499ba50e799cb1a99878f3cf3f8383b7313e2579c5951346682b5b657c9e93c7b25266edd713c95c97a40f1041f5d5074b160447c487d326b967f725c851b0ffb76b83dbb66057cbe8ef046e2d274aacd4db5324d2b4f5c7478aa15b589dc37e6ec22c30e15b7d16c55e1a27f4571944

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteA: B:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteC:

**Prueba 3**

Llave A:

1dd354b62fe5c2623f406b88a0c54f277f07356d586f3a4c4c00c8acfb6c266721e90a10677bec6cac75d79d8fb2da60a1ceda392a097a7e729fbc16508da5e437034393fc83f92bbb1fdae168d14e7b3af8000cffc4a75eb2b5883d6aa0ca64688bda010234f47d1557e2c826365d17916db6d20dc69de154743b77fb59b2c7

Llave B:

634e9df4cc30d6bce9db28d9097f4b1f4e6a146d769ef7f52f12dfd484e3d364c16d03269fc070ceffcf06851fd86eeb93bc579c07f0867258e255b2ffc8b6d87b1c3662e4c28838e292242cb4ebf1b18aa983fa5211b1d9ecd3a368447a294cb3d23bd297c1d9edffd0194d06a04f1b35afa38b5229a45a35b5cfcb67f3d710

Llave C:

8d698fd1c24fa154e647eb7fb008fa0e54c7ea84edb26dbef81d7119e36da6222ecc385c6173c345fdd1dae738537570cd0fe7b01a0b38f9bfad60dad2f2d64d48e728c5ee02fa1947ced052cae55c761335efa7daa73aacb61f7bb6121af9ebe4a5f49f9c97ddc976467674072b82ba219dd7748db6c7f5ba27549e87c8d821

Llave AB:

1dd4040a8382d109c6d97e6ef2e08c71c9410c0f9006ae92aa19b0a0747bf56c62b7f44fb62a7b1d4ee3e0bc62eefd87106f30b1795ba3cdece85f295273cf3b6098ef761c063f8190c01bcdfdd3aa8466c820d7e943c810a2fc9250340dc9ff4c92a4f3f8f485b1b8b6dd75d0b04b070fa7e20e3d1ceb758d0a8903aa2baf9e

Llave BC:

46b8348485043d88ce6fd4f4491526279eee0da91b899edcaca47fcc0f726e4d84d8ad97c63a8cd561dabf78eaa151466197bc4ea096d9b7601754a40f9132b052c46f2f975f8b98d9a01c6df9f0655da38e8396d5955c0d19857b4b497d9ff2db52d314baa6be5928f113f9a4069e58d5ef9a9d19baf5e56c406c5b95136277

Llave CA:

9242a2a740644d37bc2c341a0b68a95ae7fda88b97e71e5750d7f72786b1f3b9495a251cb7bd5e23bd9da9293b3e83103bb3627baae61d31693b7a809ad8093e5f3f686d71b0f0af95eaa97761a0af973b9e6de60bfa84615dd0eb3d8f01ce14e0fcc7650d10285ac32cbf658152e55c29c8c817053c5e70e8cc83d28ccf8761

Secreto compartido:

63b5a63c7450f225f4fd939f3b3dee66ab87266a61e3d74b56492952fa0d065b1980179c7712bd4469140a8a5baaa2665705750c97cfeec5b19a8d15fecc51a5580380ce40012b5568c63ceb9f0eeeaa161ff3121f38206d43495f76ac7ef07e20088e56b038bd614cb2d9fa4969af1421e94325de78c3e261582940d73f7ba1

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteA: B:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteC:

**Codigo fuente**

from tkinter import \*

from tkinter import filedialog

from tkinter import messagebox

from tinyec import registry

import secrets

from tinyec import ec

global PrivKey1

global PubKey1

global curve

def compress(pubKey):

    x\_hex = hex(pubKey.x)[2:]

    y\_hex = hex(pubKey.y)[2:]

    compressed = x\_hex + y\_hex

    return compressed

def decompress(compressedPubKey):

    x = int(compressedPubKey[:128], 16)

    y = int(compressedPubKey[128:], 16)

    return x, y

def FirstKey(): #Generates public and private keys

    global PrivKey1

    global PubKey1

    global curve

    curve = registry.get\_curve('brainpoolP512r1')

    # Generation of private and public key of the first user

    PrivKey1 = secrets.randbelow(curve.field.n)

    PubKey1 = PrivKey1 \* curve.g

    # Ensure the public key is compressed correctly

    PubKey1\_compressed = compress(PubKey1)

    while len(PubKey1\_compressed) != 256:

        PrivKey1 = secrets.randbelow(curve.field.n)

        PubKey1 = PrivKey1 \* curve.g

        PubKey1\_compressed = compress(PubKey1)

    # File name

    file\_name = "Llave\_publica.txt"

    # Write the compressed variable to the file

    with open(file\_name, "w") as file:

        file.write(PubKey1\_compressed)

    messagebox.showinfo("Success", "Public key saved successfully")

def SecondKey(): #Generates the second key to send

    global curve

    global PrivKey1

    compressedPubKey = str(Kc.get())

    x, y = decompress(compressedPubKey)

    public2 = ec.Point(curve, x, y)

    # Calculation of the shared key 1

    first\_SharedKey = PrivKey1 \* public2

    KeyShared1 = compress(first\_SharedKey)

    # File name

    file\_name = "Llave\_compartida.txt"

    # Write the compressed variable to the file

    with open(file\_name, "w") as file:

        file.write(KeyShared1)

    messagebox.showinfo("Success", "Shared key saved successfully")

def ThirdKey(): #Generates the final key for Diffie-Hellman

    global curve

    global PrivKey1

    compressedPubKey = str(Kcb.get())

    x, y = decompress(compressedPubKey)

    public3 = ec.Point(curve, x, y)

    # Calculation of the shared key 1

    first\_SharedKey = PrivKey1 \* public3

    KeyShared2 = compress(first\_SharedKey)

    # File name

    file\_name = "LLaveCompartidaFinal.txt"

    # Write the compressed variable to the file

    with open(file\_name, "w") as file:

        file.write(KeyShared2)

# Tkinter window

window = Tk()

window.geometry("350x300")

window.title("P5")

button1 = Button(window, text="Calculate public key", command=FirstKey)

button1.grid(row=4, column=1, ipadx=10, ipady=5)

Kc = StringVar()

entry1 = Entry(window, textvariable=Kc, width=50)

entry1.grid(row=6, column=1)

text1 = Label(window, text="K1:")

text1.grid(row=6, column=0, padx=5, pady=1)

button2 = Button(window, text="Calculate K2", command=SecondKey)

button2.grid(row=7, column=1, ipadx=10, ipady=5)

Kcb = StringVar()

entry2 = Entry(window, textvariable=Kcb, width=50)

entry2.grid(row=9, column=1)

text2 = Label(window, text="K2:")

text2.grid(row=9, column=0, padx=5, pady=1)

button3 = Button(window, text="Generate shared key", command=ThirdKey)

button3.grid(row=10, column=1, ipadx=10, ipady=5)

window.mainloop()

**Conclusión**

En conclusión, esta implementación de intercambio de claves Diffie-Hellman mediante criptografía de curva elíptica ofrece una forma segura y eficiente de establecer una clave compartida entre dos partes sin necesidad de intercambiar la clave directamente. La utilización de curvas elípticas proporciona una mayor seguridad y eficiencia en comparación con otros métodos de intercambio de claves. Además, la retroalimentación al usuario a través de mensajes informativos contribuye a una experiencia de usuario más satisfactoria. En resumen, esta implementación brinda una solución versátil y sólida para el intercambio seguro de claves en aplicaciones que requieren comunicaciones seguras y confiables.