**Escenario 1: Intercambio de Llaves Diffie-Hellman sobre un Grupo Cíclico Fp∗\mathbb{F}\_p^\*Fp∗​**

**1. Implementación del Protocolo:**

Para este escenario, implementamos el protocolo de intercambio de llaves **Diffie-Hellman** sobre un grupo cíclico multiplicativo Fp∗\mathbb{F}\_p^\*Fp∗​. La generación de las claves se realiza utilizando un número primo grande ppp y un generador del grupo ggg. Cada parte (cliente y servidor) genera una clave privada, y a partir de ella calcula su clave pública. Ambas partes intercambian las claves públicas para luego derivar un secreto compartido.

Una vez que las partes tienen el secreto compartido, este se utiliza como entrada para una función de derivación de claves (KDF). En este caso, utilizamos **scrypt**, una KDF robusta que nos proporciona una clave simétrica a partir del secreto compartido. Dicha clave es utilizada para cifrar los mensajes entre el cliente y el servidor utilizando el algoritmo de cifrado **Salsa20**.

El siguiente código muestra cómo se realiza la implementación del intercambio de llaves y el cifrado de los mensajes:

# Intercambio de llaves Diffie-Hellman

secreto\_compartido\_cliente = pow(clave\_publica\_servidor, clave\_privada\_cliente, p)

llave\_simetrica = scrypt(str(secreto\_compartido\_cliente).encode(), b'sal', 32, N=214, r=8, p=1)

# Cifrado del mensaje con Salsa20

cipher = Salsa20.new(key=llave\_simetrica)

mensaje\_cifrado = cipher.nonce + cipher.encrypt(b"Mensaje secreto del cliente")

**2. Ataque de Interceptación:**

En la segunda parte del escenario, se nos pide realizar un ataque de interceptación para intentar resolver el problema del **logaritmo discreto**. Para ello, implementamos el algoritmo **"Baby-step, Giant-step"**, que nos permite atacar el intercambio de llaves y recuperar el valor de la clave privada a partir de las claves públicas. Este método divide el problema en dos partes: la construcción de una tabla de pasos "pequeños" y la búsqueda de coincidencias en los pasos "gigantes".

Si el atacante logra resolver el logaritmo discreto, puede derivar la clave simétrica y descifrar los mensajes cifrados con Salsa20. El siguiente fragmento de código muestra cómo se realiza el ataque utilizando el algoritmo "Baby-step, Giant-step":

# Algoritmo Baby-step, Giant-step para resolver el logaritmo discreto

x = baby\_step\_giant\_step(g, clave\_publica\_servidor, p)

if x is not None:

print(f"El logaritmo discreto calculado es: {x}")

**Resultados:**

En este escenario, logramos realizar el intercambio de llaves de manera exitosa y utilizar el secreto compartido para cifrar y descifrar los mensajes de forma segura. Sin embargo, si el atacante dispone de suficiente tiempo y recursos computacionales, puede utilizar el ataque "Baby-step, Giant-step" para resolver el logaritmo discreto y comprometer la seguridad del intercambio de llaves.

**Conclusión:**

El protocolo **Diffie-Hellman** sobre un grupo cíclico Fp∗\mathbb{F}\_p^\*Fp∗​ es una técnica efectiva para generar un secreto compartido en un canal inseguro. Sin embargo, la seguridad de este método depende en gran medida del tamaño del número primo ppp y de la dificultad computacional del problema del logaritmo discreto. El uso de una **KDF** como **scrypt** y un cifrador seguro como **Salsa20** ayuda a proteger la comunicación, pero es fundamental elegir parámetros adecuados para mitigar los ataques basados en el logaritmo discreto.

1. Principales Aprendizajes en Relación con la Seguridad y la Eficiencia de los Esquemas Criptográficos Implementados:

Durante la implementación del protocolo Diffie-Hellman sobre un grupo cíclico \( \mathbb{F}\_p^\* \), pudimos observar que este esquema criptográfico es eficiente para generar un secreto compartido entre dos partes, incluso en un canal inseguro. Sin embargo, la seguridad de este protocolo depende fuertemente de la dificultad para resolver el problema del logaritmo discreto, el cual se ve influenciado por el tamaño del número primo \( p \). Cuanto mayor sea el valor de \( p \), mayor será la seguridad, pero esto también aumenta la complejidad computacional y el tiempo necesario para realizar las operaciones de intercambio.

Además, al incorporar una KDF (como scrypt) para derivar la clave simétrica, observamos una capa adicional de seguridad en el manejo del secreto compartido. En cuanto a la eficiencia de los algoritmos de cifrado simétrico como Salsa20, son significativamente más rápidos que los algoritmos asimétricos. Esto se debe a la menor carga computacional requerida para cifrar y descifrar grandes volúmenes de datos.

No obstante, el ataque basado en el algoritmo "Baby-step, Giant-step" demostró que, si los parámetros de seguridad (como el tamaño de \( p \)) no son suficientemente grandes, existe un riesgo real de comprometer el protocolo.

2. Relevancia de la Correcta Implementación de Protocolos Criptográficos y de los Posibles Ataques:

La correcta implementación de los protocolos criptográficos es esencial para asegurar la privacidad y confidencialidad de la información. Durante el intercambio de llaves Diffie-Hellman, es crucial que ambos participantes verifiquen la autenticidad de las claves públicas que están recibiendo. De no ser así, el protocolo se vuelve vulnerable a ataques como el Hombre en el Medio (MitM), donde un atacante puede interceptar las claves públicas, reemplazarlas con las suyas y establecer una conexión falsa con cada una de las partes. Este tipo de ataque pone en peligro la confidencialidad de los mensajes cifrados, ya que el atacante podría descifrar y modificar la información antes de reenviarla.

Asimismo, la implementación adecuada de una KDF asegura que la clave derivada a partir del secreto compartido sea resistente a ataques como los de fuerza bruta. Si no se siguen las mejores prácticas en la implementación de estos esquemas, los sistemas pueden quedar expuestos a vulnerabilidades que podrían haberse mitigado fácilmente.

3. Recomendaciones para un Intercambio Seguro de Llaves y una Comunicación Cifrada Robusta:

1. Utilización de Tamaños de Clave Adecuados: Es fundamental que los parámetros de seguridad, como el tamaño de \( p \) en Diffie-Hellman, sean lo suficientemente grandes para evitar ataques basados en el logaritmo discreto. Se recomienda utilizar números primos de al menos 2048 bits en aplicaciones modernas.

2. Autenticación de las Claves Públicas: Para prevenir ataques MitM, se debe incorporar un mecanismo de autenticación de claves públicas, como el uso de certificados digitales o firmas. Esto garantiza que las partes involucradas en el intercambio de llaves puedan confiar en la legitimidad de las claves públicas que reciben.

3. Derivación de Claves Seguras: La utilización de KDFs robustas como scrypt, que incorporan sal y factores de costo (como el tiempo de cómputo y la memoria), es clave para asegurar que las claves derivadas sean resistentes a ataques de diccionario o fuerza bruta.

4. Monitoreo y Prevención de Ataques: Es recomendable utilizar herramientas de monitoreo de red como Wireshark para detectar posibles ataques de interceptación. Adicionalmente, los sistemas deberían incluir medidas proactivas de prevención de ataques, como el uso de Perfect Forward Secrecy (PFS), que garantiza que la clave de sesión no se vea comprometida incluso si se compromete una clave privada.

Siguiendo estas recomendaciones y asegurando una correcta implementación de los protocolos criptográficos, se puede garantizar un intercambio de llaves seguro y una comunicación cifrada robusta, manteniendo la confidencialidad e integridad de los datos.