Universidad del Rosario
Noviembre 24, 2022
Introduccion a la Criptografia
Elgamal

Implementation Elgamal

Giancarlo González y Santiago Diaz

ABSTRACT

This project shows the implementation of the Elgamal communication channel, this implementation will be done with the help of sockets to establish bidirectional communication between both parties.

Keywords: Criptosistema, Elgamal, Python, Sockets, Firma digital

1 INTRODUCCIÓN

Desde la antigüedad se ha tenido la necesidad de proteger información de personas no autorizadas, por lo que se desarrollaron e implementaron tipos de cifrados como el cifrado cesar. En la actualidad cifrar de este modo sería algo inviable, debido a las capacidades de cómputo con las que dispone la mayoría de la población.
Hoy en día, se implementan criptosistemas más sofisticados para proteger la información, además dichos criptosistemas garantizan algunas características en la comunicación como la autenticidad y el no repudio dependiendo como se implementen.
Elgamal es un esquema de cifrado basado en el problema matemático del logaritmo discreto que consiste en encontrar un número x tal que $g^x = y$ donde g es un elemento del grupo cíclico finito G. La seguridad de Elgamal radica en dicho problema matemático, el cual, hasta ahora, con los recursos computacionales disponibles se hace poco eficiente encontrar dicho x, pues en la implementación se usan números primos muy grandes, y la cardinalidad del grupo es de al menos 2^160. Elgamal es un algoritmo de criptografía asimétrica basado en la idea de Diffie-Hellman para el establecimiento e intercambio de llaves.

2 DESARROLLO

Se implementó un canal de comunicación bidireccional asegurado mediante el método de cifrado Elgamal, dicho canal se realizó en Python con la ayuda de sockets. Adicionalmente, se implementó la firma digital basada en Elgamal para garantizar autenticidad y no repudio.
Se crea un archivo para el servidor, el cual hace la conexión al local-host por el puerto 55555 (Esto para efectos prácticos del ejemplo, se puede cambiar la dirección IP si se desea montar de manera remota.) y estará a la escucha por el mismo esperando las peticiones por parte de los clientes. Se crean dos archivos con extensión py uno para cada usuario, los cuales se conectan al local-host por el puerto 55555 y así se establece la conexión entre las dos partes.

2.1 Cifrado Elgamal

Receptor

1. Se calcula un número p de 256 bits y sé verífica que sea primo con el test de Miller-Rabin.
2. Se halla un generador α del grupo conformado por Z_p
3. Se escoge la llave privada d entre [2,p-2]
4. Se calcula la llave pública $\beta = \alpha^d mod p$

Emisor

1. Escoge $i \in$ [2, p-2]
2. Se calcula la llave efímera $k_e = \alpha^i mod p$
3. Se calcula la llave de enmascaramiento $k_m = \beta^i mod p$
4. Se encripta $y = xk_m mod p$ donde x es el mensaje.

Desencriptar Receptor

1. Calcular llave de enmascaramiento $k_m = k_e^d mod p$
2. Calcular $K_m^{-1} mod p$
3. Desencriptar haciendo $x = yK_m^{-1} mod p$

2.2 Firma digital

Generación

1. Escoger, una llave efímera $k_e \in$ [0,p-2] tal que gcd(k_e, p-1)=1
2. Calcular r = α^{k_e} mod p
3. Calcular s = (x-dr)k_e^{-1} mod p-1

Verificación

1. Calcular t = $\beta^r r^s$ mod p
2. si t = α^x mod p entonces la firma es válida

Dado que el algoritmo de Elgamal hace el cifrado mediante operaciones matemáticas, el texto deberá ser representado mediante dígitos por lo que una vez se contiene el texto plano se hace una codificación en utf-16 y posteriormente a decimal, es sobre este último sobre el que se hace el cifrado 'caracter' a 'caracter' y se concatena con la llave efímera para saber en el proceso de descifrado que valor le corresponde a cada caracter. En cuanto al descifrado se hace con la operación antes mencionada, descifrando el mensaje y luego decodificándolo en utf-16 para posteriormente mostrarlo en pantalla.

3 INSTRUCCIONES

Adjunto a este documento, se encuentran algunos archivos para realizar todos estos procesos de demostración. Para realizar algunas pruebas a continuación se encuentran unos pasos a seguir.

3.1 Inicialización de llaves.

Dentro de los documentos, existe un código de python bajo el nombre de *findpubkeys.py*. Este archivo al ejecutarse dentro de una terminal, modifica el archivo de texto llamado *llaves.txt*. Esto hará que se modifiquen las llaves publicas y en la terminal devolverá dos llaves privadas. La primera corresponde al archivo *cliente1.py* y la segunda al *cliente2.py*. Ambas llaves deben ser asignadas a sus respectivos clientes bajo la variable *d* dentro de los códigos.

3.2 Iniciación del canal

Primero que todo antes de iniciar cualquier tipo de comunicación dentro del canal, se debe realizar una iniciación al servidor que es quien va a dirigir la información. Cabe resaltar que aunque el servidor reciba los mensajes, este no puede leerlo puesto que todo queda cifrado desde el lado del cliente. Para iniciar el servidor se debe ejecutar el archivo *server.py*

Luego se procede a iniciar el cliente 1 y el cliente 2, cada uno pedirá ingresar un nombre de usuario. A partir de aquí ya se pueden enviar mensajes entre ambos clientes. Puede verificar que el canal puede enviar textos largos. Únicamente se limita a los bytes que el canal este esperando a recibir, si esto es un limitante pues simplemente se aumenta dentro del código, por lo tanto no afectaría en realidad la longitud del texto. A pesar de esto, se seleccionó un valor bastante alto para no generar conflictos.

3.3 Verificación de seguridad.

Para poder verificar que el canal es completamente seguro, se realizó un tercer cliente que va a simular algún intruso que haya en el canal. Para esto, simplemente se copio el cliente 2 variando únicamente la llave privada, puesto que el intruso no va tener acceso a esta información.

A partir de aquí, el cliente 3 ya hará parte del canal de comunicación. Para verificar primero un cifrado correcto, intente enviar mensajes desde el cliente 1, esto va provocar que a pesar que el cliente 3 puedo verificar la autenticidad de los mensajes, este no pueda leerlos, puesto que no están destinados para el y necesita la llave privada del cliente 2.

Por ultimo, si desea verificar el funcionamiento correcto de las firmas digitales, intente enviar un mensaje desde la terminal que ejecuta el cliente 3, con esto podrá notar que ninguno de los otros dos clientes va reaccionar al mensaje enviado, puesto que al verificar la firma, estos simplemente van a descartar el mensaje al no provenir del cliente deseado.

4 RESULTADOS Y DISCUSIÓN

Se implementa correctamente el canal de comunicación levantando primero el servidor, al cual se conectarán los usuarios, se ejecuta un archivo que es el encargado de generar un número primo y las llaves públicas y privadas de cada uno de los usuarios, las llaves públicas se escriben en una archivo de texto para que puedan ser consultadas por todos los usuarios, después de que son generadas todas las llaves necesarias los usuarios podrán conectarse al servidor en el que tendrán que especificar un nombre de usuario, para que dado el caso en el que quiera comunicarse con más usuarios, estos puedan reconocer con quien están hablando, pero esto es un trabajo a futuro, ya que el programa está implementado para un canal bidireccional, es decir solo entre dos usuarios.

Una vez el usuario ingresa el mensaje que quiere enviar, este es cifrado con Elgamal como se vio anteriormente y es enviado al receptor junto con la firma digital, una vez el receptor tiene la firma esta es verificada y si es válida podrá ver el mensaje.

5 CONCLUSIONES

El cifrado de Elgamal implantado con una primo de 256 bits es bastante rápido en la ejecución de las operaciones, incluso para texto de tamaño mediano, si bien se sabe, por la introducción presentada, que un dicho número no es lo suficientemente grande como para garantizar que los mensajes no podrán ser descifrados por fuerza bruta y es

que debido a los recursos computacionales con los que disponemos se hace bastante demorado el proceso de la generación y verificación del número primo, sin mencionar el tiempo de ejecución de las operaciones para el cifrado.

6 REFERENCIAS

- 262588213843476. (s. f.). Python implementation of the Miller-Rabin Primality Test. Gist. Recuperado 4 de noviembre de 2022, de https://gist.github.com/Ayrx/5884790
- 2. Built-in Functions—Python 2.7.18 documentation. (s. f.). Recuperado 4 de noviembre de 2022, de https://docs.python.org/2/library/functions.htmlpow
- Riddle, R. (2022). Elgamal [Python]. https://github.com/RyanRiddle/elgamal/blob/19ef936d748f8d90800c296f9813cb00dbd60b95/e (Original work published 2013)
- Joshi, S. (2021, octubre 22). Calcular el inverso multiplicativo modular en Python. Delft Stack. https://www.delftstack.com/es/howto/python/mod-inverse-python/