Diffie-Hellman Key Exchange chat extension

These days all of our communication is done via internet. Internet is the most dominant channel for the information flow for a simple everyday communication to extremely sensitive national level communication. The amount of data that flows from internet is insurmountable and it’s not abnormal to think that data, important or non-important, has likely chance to end up in wrong hands. In the past, if someone wanted your personal data, he had to physically steal the data from your house or office. Now, he can break into your personal digital device or commercial servers that holds your data. These data can vary from emails, bank accounts, security information that can put you in a dangerous situation. The use of personal handheld device has made it even easier for people to steal data in various ways like phishing emails, texts, password sniffing etc.

It’s ingrained in our conscience to expect privacy for any type of circumstances. To be granted privacy does not need to be extremely sensitive information, it’s just a human thing to expect uncompromised privacy. It’s very normal for one to safeguard data that is generated from our day to day behaviors and there are many tools that can help to achieve that.

Cryptography is a field of science that studied different methods and methodology to create tools that can safeguard our asset. Although its history predates to thousands of years, its equally relevant now. In recent days, cryptography has been moved into digital environment where scholars and scientist are creating different cryptographical tools to safeguard our privacy. One of the tools is Diffie-Hellman Key Exchange protocol.

Diffie- Hellman Key Exchange: Diffie-Hellman is an algorithm used to establish a secret shared between two parties. It is primarily used as a method of exchanging cryptography keys for used in symmetric encryption algorithm like AES. This algorithm solved the dilemma of sharing secret key for use in symmetric cipher, but the only means of communication is insecure. It was a first public-key protocols conceptualized by Ralph Merkle, Whitefield Diffie and Martin Hellman.

Whitfield Diffie and Martin proposed two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a public key cryptosystem enciphering and deciphering are governed by distinct keys, E and D, such that computing from D from E is computationally infeasible (e.g. requiring 101000 instructions. The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enciphered in such a way that only the intended receiver is able to decipher it. As such, a public key cryptosystem is a multiple access cipher. A private conversation can therefore be held between any two individuals regardless of whether they have ever communicated before. Each one sends messages to the other encipher in the receiver’s public enciphering key and deciphers the messages he receives using his own secret deciphering key (Whitefield Diffie and Martin E. Hellman, New Directions in cryptography, pg.1).

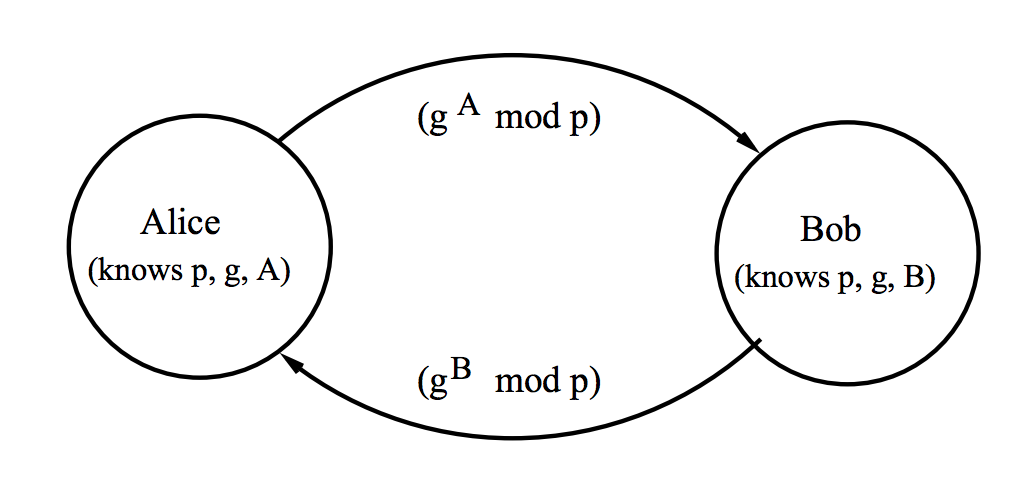


Fig 1. Diffie-Hellman Key

Steps in the Algorithm:

1. Alice and Bob agree on a prime number p and a base g.
2. Alice chooses a secret number a, and sends Bob (ga  mod p).
3. Bob chooses a secret number b, and sends Alice (gb mod p).
4. Alice computes ((gb mod p) a mod p).
5. Bob computes ((ga mod p) b mod p).

Both Alice and Bob can use this number as their key. Notice that p and g need not be protected.

Diffie-Hellman Example

1. Alice and Bob agree on p = 23 and g = 5.
2. Alice chooses a = 6 and sends 5 6 mod 23 = 8.
3. Bob chooses b = 15 and sends 515 mod 23 = 19.
4. Alice computes 19 6 mod 23 = 2
5. Bob computes 815 mod 23 = 2.

Then 2 is the shared secret.

Clearly, much larger values of a, b, and p are required. An eavesdropper cannot discover this value even if she knows p and g and can obtain each of the messages. Discovering the shared secret given g, p, g a mod p and g b mod p would take longer than the lifetime of the universe, using the best-known algorithm. This is called the discrete logarithm problem (Foundations of Computer Security).

In this application the server and client communicate with each other with sockets. A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to. Server and client use socket to establish connection. The server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request. Once the connection is established the server and client can send message to each other which emulates prototype for a chat application.

The next step is to secure the communication channel between client and server. The established channel is secured by Diffie- Hellman key exchange protocol. In Diffie- Hellman the secret key between parties is secretly agreed upon. Although the protocol is used to secure the data but the parameters used to generate secret keys between parties are shared through unsecured channel first. Once the secret keys are generated each party can send encrypted message as a ciphertext which can be decrypted by secret key.

The implementation of the application is done in java. It has four java classes; Server.java, ClientHandler.java, Client.java. ClientHandler.java is used as a utility class.

Classes and it functions

Server.java: This class functions as a server role. It opens a socket on port 5056 and waits for the connection to come through. The server keeps running on loop looking to accept requests from client. Once the request is received, the server accepts and the connection is established. The server also opens OutputStream and InputSream through which it sends and receives messages.

ClientHandler.java: This class handles thread created by server after accepting new connection. Each thread has reference to socket, input, output associated with the server to client connection. This class handles sending of parameters, secret key, and message to the client from server side. The sending and receiving of message is done through java OutputStream and InputStream.

Client. Java: This class acts as a client role in the application. It sends out connection request to the server. It also sends client parameters, messages and receives messages.

DHKey.java: This class acts as a Utility function which performs cryptographical tasks in the application. Each method in the class performs single function which are described below:

1. AliceGenerateParameters: This function generates Diffie-Hellman parameters P, G, L, keypair, and keyagreement object for the server. It also performs first phase with the server private key.
2. BobGenerateParameters: This function uses parameters sent by server to generate keypair public and key agreement object for the client. It also performs first phase of the protocol.
3. GenerateSecretkKey: This function takes encoded key from key pair and keyagreement and returns secret key. It also performs second phase of the protocol.
4. GenerateDesKey: This function takes secret key generated previously and returns DES key for the secretkey.
5. generateciphertext: This function is used to encrypt the message for each party. The secret key of the party is used to generate cipher text which is send over the channel.
6. decryptcipher: The encrypted message is passed to this function which takes the party’s secretkey to decrypt the message. The decrypted message is return as a string.

The application uses in-built java cryptography library to implement the Diffie-Hellman algorithm. The sending of the message is done using java built in input and output streams. Once the key is agreed upon by both parties, all the communication can be done in encrypted form. This application works as a prototype for establishing a shared key over insecure communication channel. It proves that two parties can agree on a key for the purpose of symmetric encryption without eavesdropper obtaining the key.

References

*Foundations of Computer Security* Retrieved from *https://www.cs.utexas.edu/~byoung/cs361/lecture52.pdf*

*New Directions in cryptography* (*6, November 1976*). Retrieved from*https://ee.stanford.edu/~hellman/publications/24.pdf*