Symmetric Key Distribution & Management

Sohan Das
Roll No. - CrS 2119

Department of Cryptology and security
Indian Statistical Institute, Kolkata

Date of the Presentation, April $13^{Th} 2023$

Overview

Introduction

Introduction

Problem Statement :

- In an office there are several users, all are connected with a common LAN-server. They want to communicate with each other in a secure manner whilst ensuring confidentiality of data being shared through that server. This can be achieved by symmetric key encryption. However how do they share the keys for the same ?

What can be a solution ?

- It's quite clear that we must to encrypt the data before sharing over the insecure network.
- Either use PKI (*Public Key Infrastructure*)
- Or, use Private Key Encryption Algorithm

Introduction

PKI has several disadvantages :

- Depends on some computationally hard problem, so usually both the encryption and decryption algorithms are too much complicated and also it requires a lots of computation power. Systems may become slow down.
- As quantum computation becomes reality, so using *Brute-force* ciphertext can be decrypted.

Can Symmetric Key Encryption give same or better security than PKI ?

- How do then the symmetric key will be shared between users ?

Let's discover some protocol which can solve our problem using Symmetric Key Encryption algorithm.

Client-Server Symmetric Key distribution Protocol

Intermediate Overview

Client-Server Symmetric Key distribution Protocol

- Proposed Protocol : Session key distribution
- Designed Protocol : Master Key generation & distribution

Proposed Protocol : Session Key Distribution

Needham-Schroeder Symmetric Key Protocol :

- It's basically a symmetric key distribution protocol with authentication.
- In an article published ACMin 1978 *Roger M. Needham*, *Michael D. Schroeder* published protocols for the use of encryption for authentication in large networks of computers[NS 78][NS 87].
- In 1981 *Dorothy E. Denning* and *Giovanni Maria Sacco* found out **replay attack** on it and they published a paper on key distribution in a computer network, suggested to use timestamp instead of nonce based authentication[DS 81].

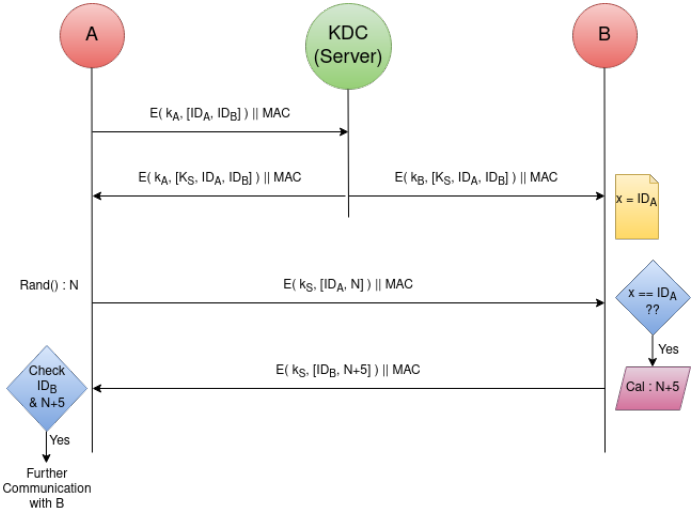
Some modifications has been done :

- Whenever one user A wants to communicate with another user B, server will provide the symmetric session key to both of them individually and user B will remain in listen mode expecting data from user A

Main Target

1. Build a client-server application through which users, connected with the same network, can share data among them, maintaining the confidentiality of the data being shared over the network.
2. I have chosen symmetric key encryption algorithm for secure communication between users over the network.
3. So some symmetric key distribution algorithm is needed. The basic idea is taken from *Needham-Schroeder symmetric key distribution protocol*[SPORE]
4. In the above protocol it is assumed that every user shared a private key with the server from early. We call it master key for every user, which should be shared before starting the above protocol.
5. We have to design a protocol to generate that master key. Basically it's also a symmetric key distribution algorithm.

Modified *Needham-Schroeder Symmetric Key Distribution Protocol*



Assumptions for the above session key generation and distribution protocol

1. Each user shares a secret key with the server. User A shares his secret key k_A only with the server. We call it master key of user A.
2. The server is a trusted server.
3. Server has the access of **QRNG**, so that it can generate as many random numbers as required and those will be used for session keys.
4. **AES-256** will be used as for encryption scheme. Also Will follow **AEAD** for authentication purpose & **MAC** (after encryption) for checking the integrity.

Intermediate Overview

Client-Server Symmetric Key distribution Protocol

- Proposed Protocol : Session key distribution
- Designed Protocol : Master Key generation & distribution

Design of Master Key Generation Protocol

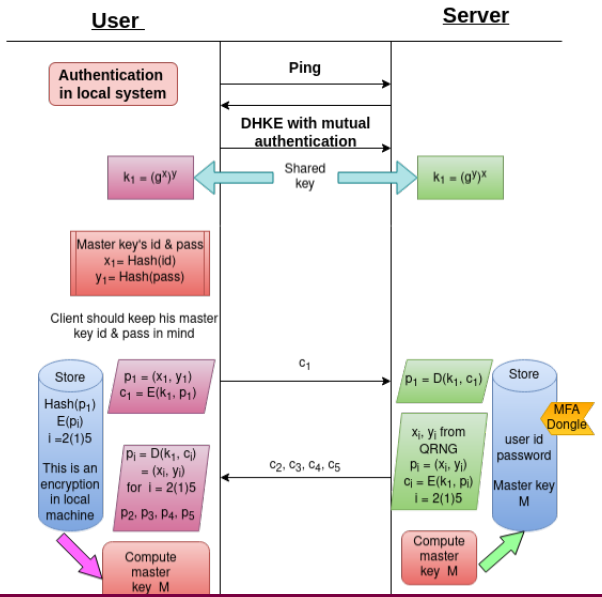
Basic Overview

1. Using some key exchange protocol (like **Diffie-Hellman** or **ECDH**) with mutual authentication establish a shared key, $\boldsymbol{k_1}$ (say).
2. User will choose his two secrets 1. *master key id* , 2. *master key password*
3. Using *SHA-256* those two secrets will be hashed and let's make a order pair of those two hashed secrets.
4. Let's use AES-256 (with CBC) and $\boldsymbol{k_1}$ as key for encryption and user will send the encrypted order pair to the server.
5. Server will choose 8, 256-bit long, random numbers from it's *QRNG*, will make 4 order pairs and then the server will send those 4 order pairs back to the user in encrypted form.

Design of Master Key Generation Protocol

Basic Overview

6. Now both the user and the server have 5 order pairs. Let's consider these as 5 points in the plane of finite fields, $(Z_p, \oplus_p, \odot_p) \times (Z_p, \oplus_p, \odot_p)$.
7. Use ***Lagrange Interpolation over finite fields*** and generate the unique polynomial $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$ of degree less than or equal to 4, where $a_i \in (Z_p, \oplus_p, \odot_p), \forall i \in \{0, 1, 2, 3, 4\}$
8. Select some value $b \in (Z_p, \oplus_p, \odot_p)$ by mutual discussion.
9. Set $f(b)$ as master key for user A. We chose $f(0) = a_0$ as the master key for user A
10. Master key generation process is done. Server will store it in it's database securely.
11. User will store only the $hash(f(b))$ and those 4 points received from server securely.


User
Server
Authentication in local system
Ping
DHKE with mutual authentication
$k_1 = (g^x)^y$
Shared key
$k_1 = (g^y)^x$
Master key's id & pass
x_1= Hash(id)
y_1= Hash(pass)
Client should keep his master key id & pass in mind
Store
Hash(p_1)
E(p_i)
i =2(1)5
This is an encryption in local machine
$p_1 = (x_1, y_1)$
$c_1 = E(k_1, p_1)$
c_1
$p_1 = D(k_1, c_1)$
Store
MFA Dongle
x_i, y_i from QRNG
$p_i = (x_i, y_i)$
$c_i = E(k_1, p_i)$
i = 2(1)5
user id
password
Master key M
$p_i = D(k_1, c_i)$
$= (x_i, y_i)$
for i = 2(1)5
p_2, p_3, p_4, p_5
c_2, c_3, c_4, c_5
Compute master key M
Compute master key M

Implementation

Intermediate Overview

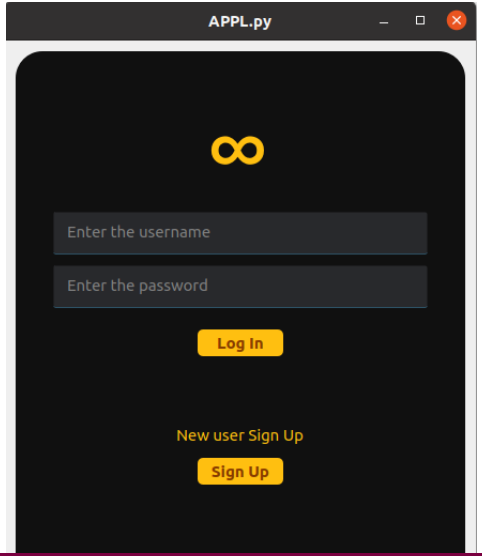
- Implementation
  - Client / user side application
    - Interface of the application
  - Server side application
    - Server set up & maintenance
  - Progress Report

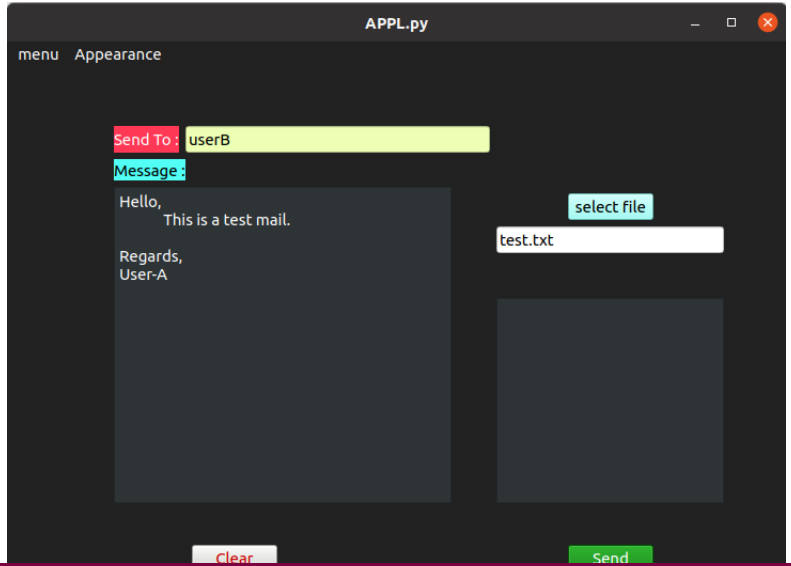
Requirements and used tools :

- *PyQt5* is used to implement the GUI of the client side application. I used *SQLite3* to create local database.
- *Python3* is preferred to write the back end codes like codes for *Lagrange interpolation method*, codes for sending messages, codes for connection with the database etc.
- This application provides user an interface where user can type messages or attach files to send to another user over the local network in a secure manner using the above discussed two protocols.
- I used *pycryptodome*, a *Python* library for computing *SHA-256* & *AES-256*. I would like to implement those latter.

Log In Window

1. This page will be opened first
2. Users have to put correct credentials to be logged in
3. Then the Main Window will be opened
4. for non existing users there is *Sign UP* button.
5. Exactly one user for one system will be preferred.





1. BY clicking the *send* button master key id & password window will be opened.
2. *Clear* button clears all the fields
3. Using *select file* button user can select any file, wanna send to another user

Master Key ID & Password Window

1. By clicking on *submit* button hashed value will be matched with the saved ones in the local database.
2. Then using *Lagrange Interpolation method over finite fields* the master key will be computed
3. Connection with the server will be made and *Session Key generation protocol* will be followed to get session key.
4. Then the typed message and selected file will be encrypted with the session key and sent to user B.

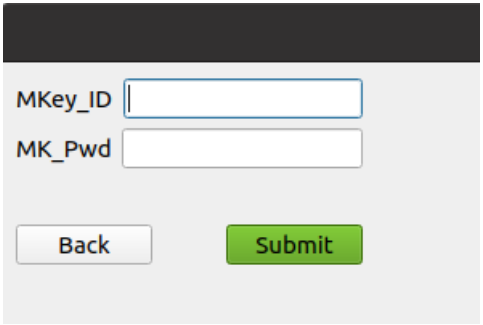
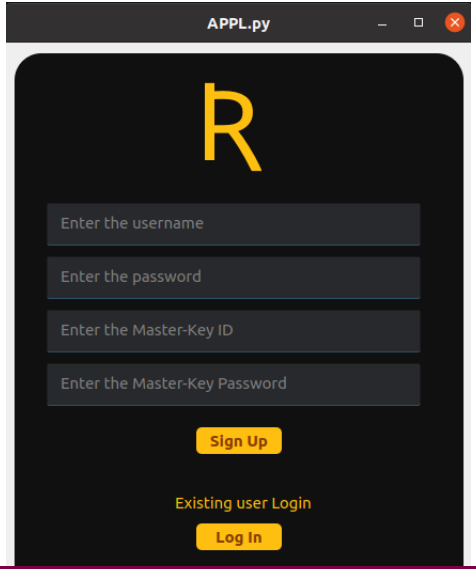


Figure: Master Key ID & Password Window

1. First time a user have to be signed up by filling this form
2. This is called registration process, which will be going to be happened only once by following the *Master Key generation protocol*



Intermediate Overview

- Implementation
  - Client / user side application
    - Interface of the application
  - Server side application
    - Server set up & maintenance
  - Progress Report

Server set up & maintenance

Architecture & Requirements

Basically three types of server are required

- Mail Server
- Storage or database server
- Authentication server

After configuring these servers, connection between server and client side application must be established.
For providing better experience to users we must have to balance load on the server and to handle traffic, some load balancer cluster or gateway may be required.

Overview of the connection of server & client side application

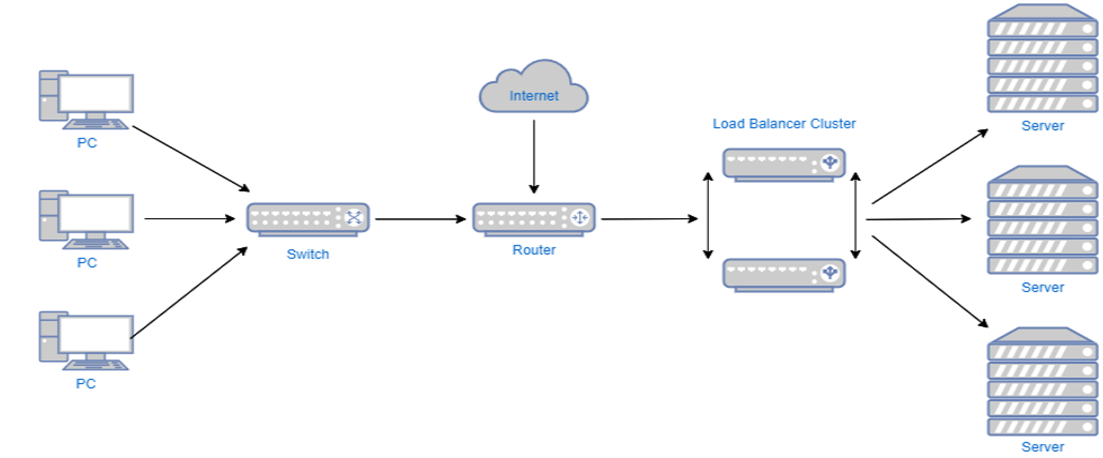


Figure: Client-Server communication

Intermediate Overview

- Implementation
  - Client / user side application
    - Interface of the application
  - Server side application
    - Server set up & maintenance
  - Progress Report

Progress Report

Task	Sub task	Status	Remarks
Protocol Design	Session Key Distribution	Done (100%)	modified
	Master Key Distribution	Done (100%)	designed
Client Side App	GUI	Done(100%)	PyQt5
	Master Key gen Algo	Implemented (80%)	Python3
Server Side App	Web Application	In-Progress (20%)	20.04.23
	Configuring servers	Have to start	30.04.23
Connect Server & User Side App	Session Key gen Algo	Will implement	15.05.23
	Security analysis & others	Start soon	30.05.23

Conclusion

Conclusion

Work done :

- Key distribution protocols, both the session key and master key distribution protocols are designed
- Client/ user side application has been built
- Master key generation algorithm is implemented using local database

Future Plan:

- Configure required servers
- Implement Session key distribution algorithm
- Connect user's app with the server, do security analysis and check performances

Citations I

[NS 78] Michael D. Schroeder [NS] Roger M. Needham. “Using encryption for authentication in large networks of computers”. In: 21 (12 Dec. 1978). Ed. by Robert L. Ashenhurst. https://dl.acm.org/doi/pdf/10.1145/359657.359659. ISSN: 0001-0782. DOI: 10.1145/359657. URL: https://dl.acm.org/doi/10.1145/359657.359659.

[DS 81] Giovanni Maria Sacco [DS] Dorothy E. Denning. “Timestamps in key distribution protocols”. In: 24 (8 Aug. 1981). Ed. by Robert L. Ashenhurst. https://dl.acm.org/doi/pdf/10.1145/358722.358740. ISSN: 0001-0782. DOI: 10.1145/358722. URL: https://dl.acm.org/doi/10.1145/358722.358740.

Citations II

[NS 87] Michael D. Schroeder [NS] Roger M. Needham. “ACM SIGOPS Operating Systems Review”. In: 21 (1 Jan. 1987). Ed. by William M Waite. https://dl.acm.org/doi/pdf/10.1145/24592.24593. ISSN: 0163-5980. DOI: 10.1145/24592. URL: https://dl.acm.org/doi/10.1145/24592.24593.

[SPORE] Michael Schroeder [SPORE] Roger Needham. “Needham Schroeder Symmetric Key”. In: (RE). URL: http://www.lsv.fr/Software/spore/nssk.html.

Thanks for your attention!

Are there questions?