**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO BÀI TẬP LỚN MÔN CÁC HỆ PHÂN TÁN**

**ĐỀ TÀI:** Ứng dụng MySSH mô phỏng chương trình “ssh” để client kết nối từ xa đến server

**Gv hướng dẫn: Nguyễn Bình Minh**

**Sinh viên thực hiện:**

1. **Nguyễn Bình Minh – 20152453**
2. **Đỗ Đức cảnh – 20150314**
3. **Trịnh Thị Phương - 20152916**

**Lớp:** Các hệ phân tán - 101905

**năm học 2017 - 2018**

# Mục lục

MySSH

[Mục lục 2](#_Toc513671228)

[Lời nói đầu 2](#_Toc513671229)

[Phân tích yêu cầu 3](#_Toc513671230)

[3.1. Tổng quan 3](#_Toc513671231)

[3.3. Client 3](#_Toc513671232)

[3.3. Server 3](#_Toc513671233)

[Giới thiệu SSH 3](#_Toc513671234)

[4.1. Khái niệm 3](#_Toc513671235)

[4.2. Đặc điểm của SSH 4](#_Toc513671236)

[4.3. Cấu trúc của SSH 4](#_Toc513671237)

[4.4. Cơ chế hoạt động 4](#_Toc513671238)

[4.5. Kỹ thuật mã hóa 6](#_Toc513671239)

[1. Symmetric Encryption 6](#_Toc513671240)

[2. Asymmetric Encryption 6](#_Toc513671241)

[3. Hashing 6](#_Toc513671242)

[Phân tích Kiến trúc MySSH 7](#_Toc513671243)

[5.1. Kiến trúc Client-Server 7](#_Toc513671244)

[5.2. Sơ đồ lớp phía server 8](#_Toc513671245)

[5.3. Sơ đồ lớp phía client 9](#_Toc513671246)

[Phân tích hành vi và vấn đề trong hệ phân tán 10](#_Toc513671247)

[6.1. Communication 10](#_Toc513671248)

[6.1. Security 10](#_Toc513671249)

[6.2. Scalability 10](#_Toc513671250)

[6.3. Failure Handing 10](#_Toc513671251)

[6.4. Concurrency 11](#_Toc513671252)

[Demo MySSH 11](#_Toc513671253)

[Tài liệu tham khảo 11](#_Toc513671254)

# Lời nói đầu

Cùng với sự phát triển của web/server đòi hỏi các công cụ remote control ngày càng phải được cải tiến hơn (git , remote shell,..). Một trong những công cụ nổi tiếng phổ biến hiện nay là SSH hay Secure Shell

# Phân tích yêu cầu

## 3.1. Tổng quan

* Client kết nối từ xa đến Server thực hiện các lệnh hệ thống trên Server dùng socket, giao thức TCP
* Kịch bản hoạt động yêu cầu như sau:
  + Giao diện dòng lệnh
  + Multiclients: cho phép nhiều clients kết nối đồng thời (giới hạn N clients, vượt quá N thì không cho phép kết nối)

## 3.3. Client

* Khi client kết nối tới phải đăng nhập (username / pass). CSDL người dùng là của chương trình, không phải của hệ thống, cơ chế lưu tùy ý (CSDL, text, v.v)
* Kết thúc phiên làm việc client gửi lệnh ngắt kết nối tới server và chấm dứt kết nối.

## 3.3. Server

* Server thông báo kết nối thành công hay không
  + Khi kết nối thành công thì client có thể thực hiện các lệnh trên hệ thống của server
* Các lệnh có thể có tham số hoặc không
* Một số lệnh cơ bản: hiển thị thư mực hiện thời, in danh sách các tệp, thư mục trong thư mục hiện thời, xóa, di chuyển tệp, thư mục, hiển thị ngày giờ hệ thống, v.v...
* Server phân tích lệnh nhận được từ phía client, thực hiện và trả kết quả cho client
* Nếu các lệnh thực hiện không được phép chạy thì phải thông báo lỗi.
* Lệnh chạy trên server phụ thuộc vào hệ điều hành mà server được cài đặt lên (Windows, Linux)

# Giới thiệu SSH

## 4.1. Khái niệm

“SSH is a protocol for secure remote login and other secure network

services over an insecure network” (rfc 4251).

SSH (Secure Shell) là 1 giao thức mạng để thiết lập kết nối 1 cách bảo mật và an toàn

* Nó hoạt động theo chồng giao thức TCP/IP
* Có thể thiết lập 1 đường hầm an toàn
* SSH Client (OpenSSH,..) dùng để giao tiếp giữa client và server có sử dụng các cơ chế mã hóa để đảm bảo tiêu chí an toàn thông tin thay thế cho các phương pháp cũ về trước như Telnet , rlogin,..

## 4.2. Đặc điểm của SSH

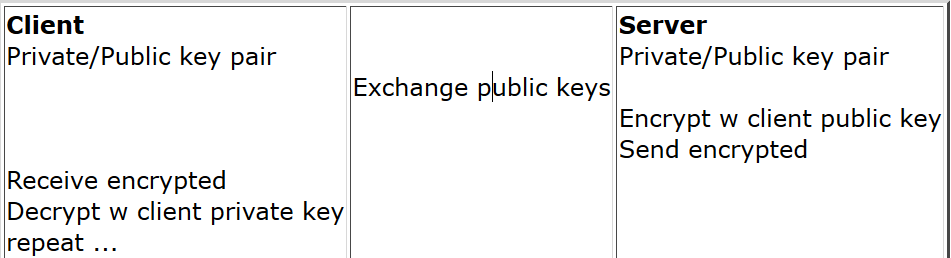
* Tính bí mật (Privacy) do sử dụng các cơ chế mã hóa mạnh mẽ
* Tính toàn vẹn (Integrity) đảm bảo tính toàn vẹn dữ liệu trên đường truyền
* Chứng minh xác thực (Authentication) sử dụng phương pháp xác thực giữa client và server
* Giấy phép (Authorization) dùng để điều khiển truy cập đến tài khoản
* Chuyển tiếp (Forwording) hoặc tạo đường hầm để mã hóa những phiên khác dựa trên giao thức TCP/IP

## 4.3. Cấu trúc của SSH

* Server
  + Một chương trình cho phép đi vào kết nối SSH với một bộ máy, trình bày xác thực, cấp phép, … Trong hầu hết SSH bổ sung của Unix thì server thường là sshd
* Client
  + Một chương trình kết nối đến SSH server và đưa ra yêu cầu như là “log me in” hoặc “copy this file”. Trong SSH1, SSH2 và OpenSSH, client chủ yếu là ssh và scp.
* Session
  + Một phiên kết nối giữa một client và một server. Nó bắt đầu sau khi client xác thực thành công đến một server và kết thúc khi kết nối chấm dứt.
* Key
  + Có hai loại khóa: khoá đối xứng ( khoá bí mật) và khoá bất đối xứng ( khóa công khai). Một khoá bất đối xứng hoặc khoá công khai có hai phần: thành phần công khai và thành phần bí mật.

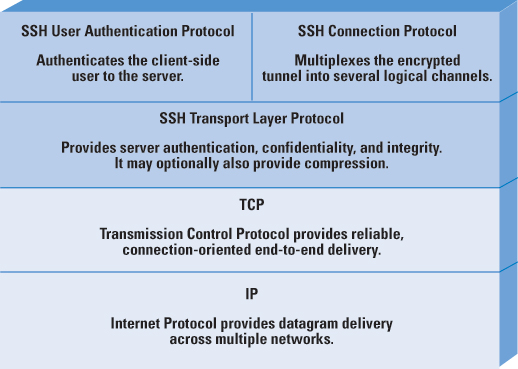
## Cơ chế hoạt động

Sơ đồ trình tự:



Một phiên làm việc SSH phải trải qua 4 bước:

* Thiết lập kết nối ban đầu (SSH-TRANS)
* Tiến hành xác thực (SSH-AUTH)
* Mở phiên kết nối để thực hiện các dịch vụ (SSH-CONN)
* Chạy các ứng dụng SSH (Có thể là SSH-SFTP, SCP)



**SSH-TRANS**: là khối xây dựng cơ bản cung cấp kết nối ban đầu, ghi chép giao thức, xác thực server, mã hóa cơ bản và bảo toàn dữ liệu. Sau khi thiết lập kết nối, client có một kết nối độc lập và bảo mật.

Sau đó, client dùng SSH-AUTH để xác thực đến server. SSH-AUTH yêu cầu một phương thức: Public key với thuật toán DSS. Ngoài ra, sử dụng mật khẩu và hostbased.

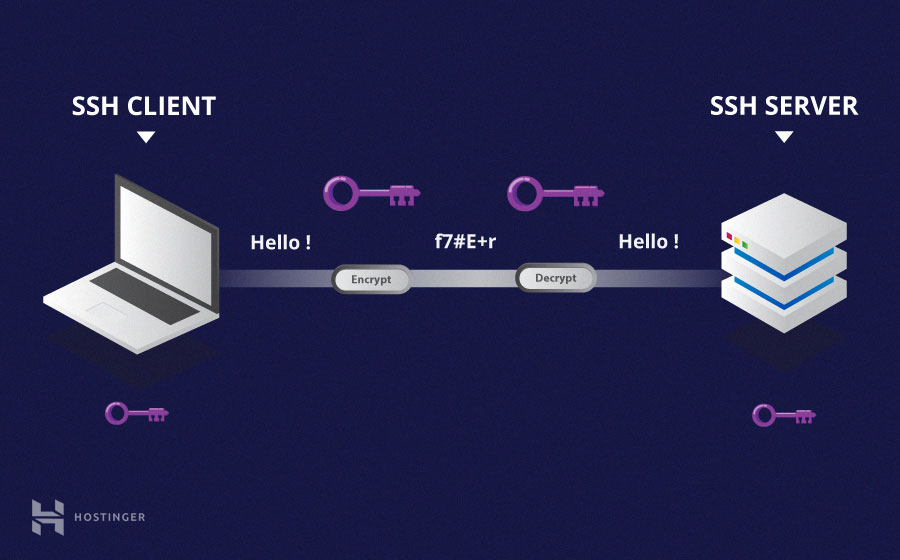
Sau khi xác thực, SSH client yêu cầu SSH-CONN để cung cấp một kênh riêng biệt qua SSH-TRANS.

Ngoài ra, còn cung cấp các dịch vụ như Remote Login and Command Execution, agent fowarding, files transfer, TCP port fowarding, X fowarding,...

Cuối cùng, một ứng dụng có thể sử dụng SSH-SFTP hoặc SCP truyền file hoặc thao tác remote từ xa.

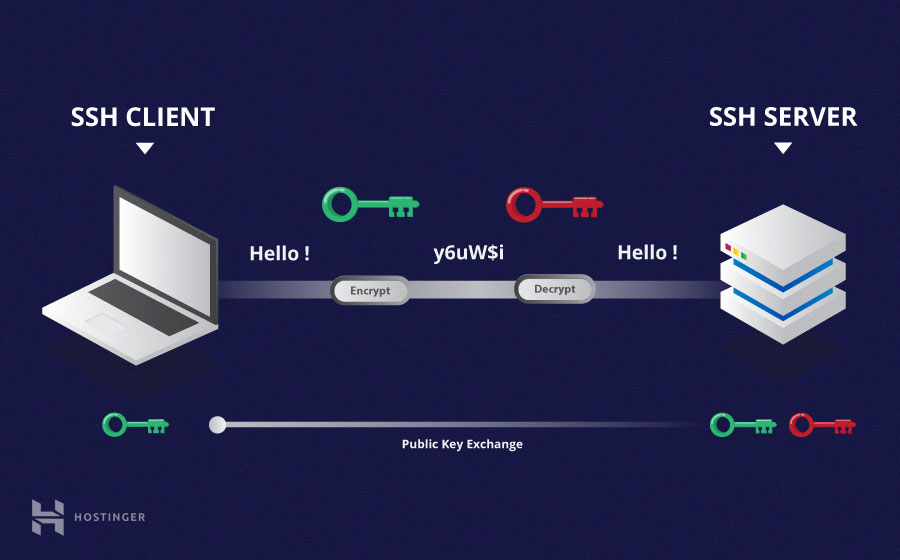
## 4.5. Kỹ thuật mã hóa

### ****Symmetric Encryption****



Symmetric keys được sử dụng để mã hóa toàn bộ liên lạc trong phiên giao dịch SSH. Cả client và server tạo chung một key bí mật như là một phương thức thỏa thuận, và key đó không được tiết lộ cho bên thứ ba. Quá trình tạo symmetric key được thực hiện bởi **key exchange algorithm**.

### ****Asymmetric Encryption****



Asymmetrical encryption không được dùng để mã hóa toàn bộ phiên SSH. Thay vào đó, nó chỉ được sử dụng trong quá trình trao đổi thuật toán của khóa của symmetric encryption. trước khi bắt đầu một phiên giao dịch an toàn, cả 2 đồng ý tạo ra một cặp public-private key tạm, chia sẽ private keys để tạo một khóa secret key chung.

### ****Hashing****

Hashing một chiều là một dạng mã hóa khác sử dụng trong Secure Shell Connections. Hash một chiều khác với cả 2 phương thức mã hóa trên ở chỗ nó không được sinh ra để giải mã. Chúng tạo ra một giá trị duy nhất với độ dài nhất định cho mỗi lần nhập liệu mà không có hướng nào khác để khai thác.

# Phân tích Kiến trúc MySSH

## 5.1. Kiến trúc Client-Server

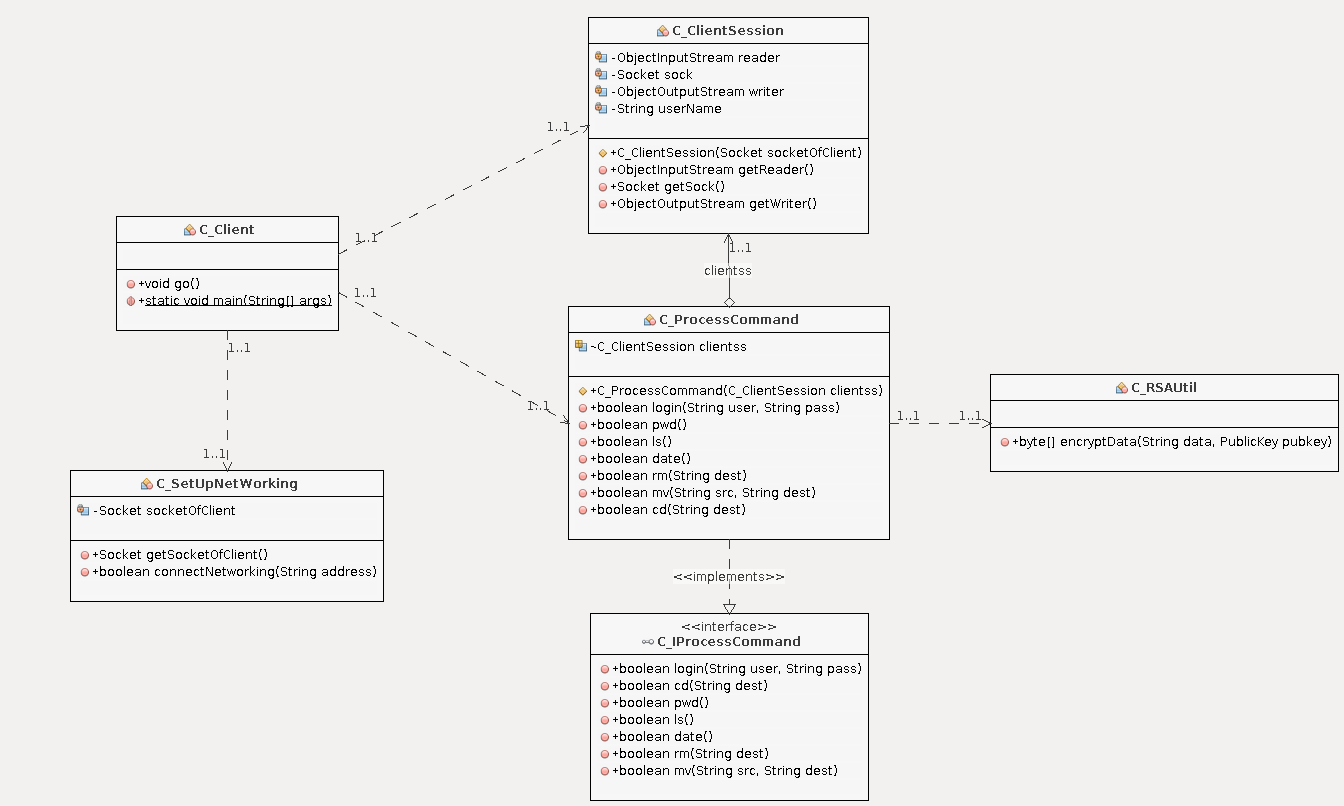
Chương trình có 2 phần là client và server

Server dùng hệ điều hành Ubuntu 18.04

Các mà nguồn ssh-server sẽ được đặt trong package server còn package clients chứa mã chương trình ssh-client

## 5.2. Sơ đồ lớp phía server

## 5.3. Sơ đồ lớp phía client



# Phân tích hành vi và vấn đề trong hệ phân tán

## 6.1. Communication

Sử dụng giao thức tầng giao vận là TCP nên cung cấp tính reliability (phía nhận gửi lại ACK cho phía gửi) .

Có 3 mô hình giao tiếp trong hệ phân tán là:

* Inter-Process Communication
* Remote Invocation
* Indirect Communication

Theo yêu cầu đề tài sử dụng Inter-Process Communication hay cụ thể là Socket API. Nhược điểm của mô hình này:

* Space coupling: client phải biết chính xác địa chỉ server (IP Address=127.0.0.1, port=5000).
* Time coupling: server luôn lắng nghe khi có client mới muốn khởi tạo kết nối cũng như lắng nghe để nhận yêu cầu các client hiện tại đang kết nối .

## 6.1. Security

1. Sử dụng cơ chế username/ password để xác thực người dùng.
2. Sử dụng cơ chế Asymmetric cụ thể là RSA-1024 để mã hóa tài khoản.

## 6.2. Scalability

Chương trình sử dụng kiến trúc client - server nên tạo ra single-point of failure tại server

## 6.3. Failure Handing

Các lỗi và khả năng chịu lỗi:

1. Lỗi phía người dùng nhập sai cú pháp ,ssh client sẽ thông báo Unrecognized input
2. Client Unable locate to server : sử dụng exception để thông báo cho người
3. Omission Failure (receive omission or send omission ) or lost message : dẫn đến hệ thống bị treo. Giải pháp dùng bộ timer cứ 3 phút kiểm tra nếu chương trình trong 5 phút không nhận được kết quả thực hiện sẽ đóng socket
4. Có cơ chế ghi log phía server
5. Reliable Communication có 2 mô hình là reliable request-reply communication và reliable group communication.

Trong đề tài sử dụng mô hình Request-reply communication nên thứ tự các thông điệp là quan trọng. Trong trường hợp số client lớn phải gửi ACK cho từng client với từng thông điệp cũng trở thành vấn đề.

## 6.4. Concurrency

Khi có client kết nối thì hệ thống tự động tạo ra 1 socket và một luồng riêng để xử lý

Đảm bảo biến CurrentConnect chia sẻ chung, có tính Consistency tức là không cho 2 luồng cùng truy cập vào bằng cách thêm synchronization vào đầu method disconnect() hay connect() . Ngoài ra đề tránh, tăng số lượng client quá max kết nối đã thiết lập ban đầu sử dụng cặp wait() và notify() bao bên ngoài.

Bên trong một số luồng sử dụng StringBuilder thay vì StringBuffer mặc dù StringBuilder là non-synchronized nhưng lại hiệu quả hơn tức là thời gian thưc hiện nhanh

# Demo MySSH

Chương trình sử dụng ngôn ngữ java được viết trên NetBeans IDE và hệ điều hành ubuntu( git: <https://github.com/BinhMinhs10/MySSH>)

Các bước chạy chương trình demo :

1. Vào package server.view chạy class Server
2. Thực hiện các yêu cầu đến khi có thông báo cài đặt mạng thành công
3. Vào package clients.view chạy class C\_Client, mỗi lần chạy file chính là 1 lần có 1 client thực hiện

# Tài liệu tham khảo

1. VanDyke Software, Inc , White paper SSH
2. Mario Cagalj, Lectures SSH protocol
3. Dr. Nguyen Binh Minh, Lecture Distributed Systems