**Báo cáo Assignment Môn Bảo mật thông tin**

**Họ và tên**: Phạm Dương Thành Long

**MSSV**: 51603190

1. **Tóm tắt công việc thực hiện:**

Theo yêu cầu của đề bài được giao em đã xây dựng thành công quá trình mã hóa thông điệp theo RSA gửi từ Client đến Server. Quá trình thực hiện được bắt đầu bằng cách chạy file BigIntServer để bắt đầu tạo public key và private key. Các key này được tạo từ file RSAKeyGen được dùng để tạo các khóa trên vụ cho quá trình mã hóa thông điệp và giao tiếp giữa 2 máy chủ. Còn về file BigIntClient được xây dựng để nhận một thông điệp từ Client mà theo như đề bài là tên người dùng. Input sau khi nhận sẽ được chuyển ngay lập tức sang dạng bigInterger và chuyển cho server. Tiếp theo client sẽ nhập lần lượt trao đổi public key và giá trị n cho nhau nếu các giá trị của 2 bên nhập vào trung nhau thông điệp sẽ được giải mã thành công bên phía server và hiển thị đầy đủ tên người dùng. Tiếp theo thì em cũng đã hoàn thành về mặt lí thuyết của thuật toán RSA sẽ được mô tả ở phần 2.

1. **Giới thiệu:**
2. ***Giải thích cơ chế hoạt động của thuật toán RSA:***

*A.1 Giới thiệu sơ lược:*

Mã hóa **RSA** đang là một trong các thuật toán mã hóa được sử dụng phổ biến hiện nay vì tốc độ và tính tiện lợi của nó. RSA là dạng mã hóa khóa công khai *(public-key encryption),* tức là RSA sử dụng 2 khóa khác nhau để mã hóa và giải mã: 1 khóa công khai *(public key)* được cung cấp mọi người và 1 khóa bí mật *(private key)* được cấp duy nhất cho người nhận *(receiver).* Bất kỳ ai cũng có thể sử dụng khóa công khai để mã hóa dữ liệu cần gửi *(sender).*

Tuy nhiên, để giải mã dữ liệu được nhận thì người nhận tin cần cung cấp khóa bí mật và chỉ có người nhận này mới biết do đó tính bảo mật sẽ nằm ở đây. Khác với mã hóa khóa đối xứng *(symmetric-key encryption),* khóa bí mật không bao giờ được truyền ra ngoài, do đó dữ liệu đã mã hóa, dù bị nghe lén từ bên thứ 3, cũng không thể bị giải mã vì không có khóa bí mật. Do vậy, RSA thường được dùng trong giao thức SSL/TSL *(kể cả HTTPS*) và giấy phép điện tử *(digital certificate).*

*A.2 Nguyên lý hoạt động*

Nguyên lý cơ bản của RSA dựa trên nhận định là có thể tìm được 3 số nguyên dương rất lớn ***e, d*** và ***n*** mà trong đó:

Lúc này, dù cho có biết cả ***e, n*** hay cả ***m*** thì cũng rất khó để tìm ra ***d***. Tiếp theo, chúng ta sẽ đi sâu vào từng công đoạn của RSA bao gồm: Mã hóa, giải mã và sinh khóa.

**Mã hóa:**

Giả sử Lộc muốn gửi cho một message ***M*** cho Phi.

Đầu tiên, chia thông điệp ***M*** thành từng phần nhỏ, mỗi phần có chức năng biểu diễn bởi một số nguyên **m** sao cho **.** Việc chuyển đổi này bắt buộc phải được thực hiện một cách ngẫu nhiên và **m** không được bao gồm các giá trị quá dễ rà soát (*ex: 0 và 1*) nhưng vẫn đảm bảo có thể suy **M** từ **m**.

Tiếp đến, Lộc sẽ tạo ra *bản mã hóa* ***c*** được xây dựng trên khóa công khai **e** của Phi như sau*: .*

Sau đó, Lộc gửi **c** cho Phi

*Lưu ý: sao khi mã hóa, chính Lộc cũng cũng không thể giải mã được từ* ***c*** *thành* ***m.***

**Giải mã:**

Sau khi nhận được từ Lộc *bản mã hóa c*. Lúc này, Phi sẽ tiến hành tìm ra m dựa trên *khóa bí mật d( private key )* được cung cấp từ Lộc theo công thức như sau:

Đến đây, Dựa vào **m** từ công thức trên, Phi có thể khôi phục lại mẫu tin ban đầu M**.**

**Sinh khóa:**

Chọn 2 số nguyên tố khác nhau ***p*** *và* ***q:***

Tìm giá trị ***n***, dựa trên công thức sau: . Độ dài của n cũng chinh là độ dài của khóa( tính theo bit – khóa thông dụng hiện nay tối thiểu có 2048 bit).

*Tính:*

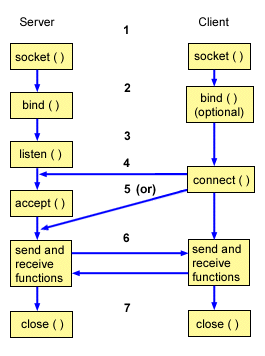
Trong đó: : phi là hàm Euler. cũng là số lượng số nguyên dương nhỏ hơn n mà nguyên tố cùng nhau với n.

Chọn một số nguyên e thỏa mãn : và .

*Tiếp theo, ta tìm* ***d****:*

1. ***Giải thích cơ chế hoạt động của Client – Server dựa trên việc sử dụng*** ***connection-oriented sockets:***

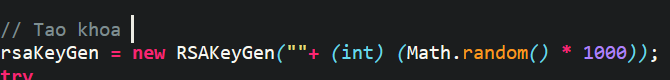
Mỗi socket có một luồng sự kiện điển hình. Trong mô hình Client – Server dựa trên connection-oriented sockets. Socket trên tuyến trình của server chờ đợi các yêu cầu từ phía client. Để thực hiên quá trình trên, Trước tiên máy chủ sẽ thiết lập một địa chỉ (address). Sau đó server sẽ chờ cho đến khi nào có client yêu cầu dịch vụ. Việc trao đổi dữ liệu giữa client và server được diễn ra chỉ khi client đã kết với server vào một socket . server lúc này sẽ thực hiện các yêu cầu của client và gửi trả lại kết quả cho client. Cụ thể các bước thực hiện được mô tả như hình sau:



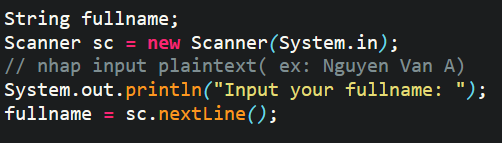
* Bước 1: API *socket ()* tạo một điểm cuối*( end point )* để liên lạc và trả về một mô tả về socket đại diện cho điểm cuối.
* Bước 2: Khi một ứng dụng có một bộ mô tả socket thì nó có thể liên kết duy nhất với duy nhất tên một socket giống với mô tả đó. Các Server phải liên kết( *bind()* ) một tên để có thể truy cập từ mạng.
* Bước 3: API *listen ()* mô tả việc sẵn sàng chấp nhận các yêu cầu kết nối của client. Khi *listen ()* được cấp cho socket, socket đó không thể chủ động bắt đầu các yêu cầu kết nối. *listen ()* được cấp sau khi socket được cấp phát với *socket ()* và *bind ()* liên kết một cái tên với socket. *listen ()* phải được giải quyết trước khi *acept ()* được ban hành.
* Bước 4: Ứng dụng của client bắt đầu quá trình *connect ()* trên luồng của socket để thiết lập kết nối với server.
* Bước 5: Server *bắt đầu accept ()* để chấp nhận yêu cầu kết nối từ client. Server lúc này phải thực hiện *bind ()* và *listen ()* thành công trước khi bắt đầu thực hiện *accept ().*
* Bước 6: Khi kết nối được thiết lập giữa các luồng socket (giữa client – server), ta có thể sử dụng bất kỳ API chuyển dữ liệu nào mà socket đó hỗ trợ. Clietn và server có nhiều API chuyển dữ liệu để chọn như : *send (), recv (), read (), write () và các API khác.*
* Bước 7: Khi muốn dừng lại quá trình tên, ta sẽ thực hiện lệnh *close()* để giải phóng các tài nguyên đang tồn tại trên socket đó.

**(3). Client Code and its Description:**

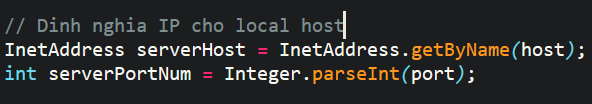
Bước đầu,tạo khóa trước khi bắt đầu mã hóa plain text

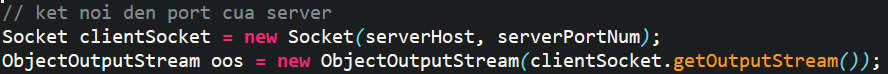


Kế tiếp, ta sẽ tiến hành tạo lệnh scanner và nhập plain text cần mã hóa:

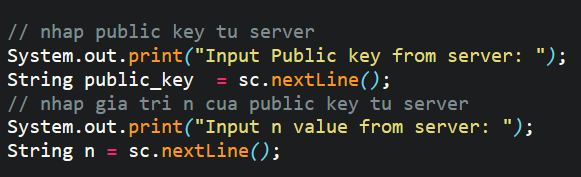


Sau đó, Định nghĩa IP cho local host được người dùng nhập và kết nối:



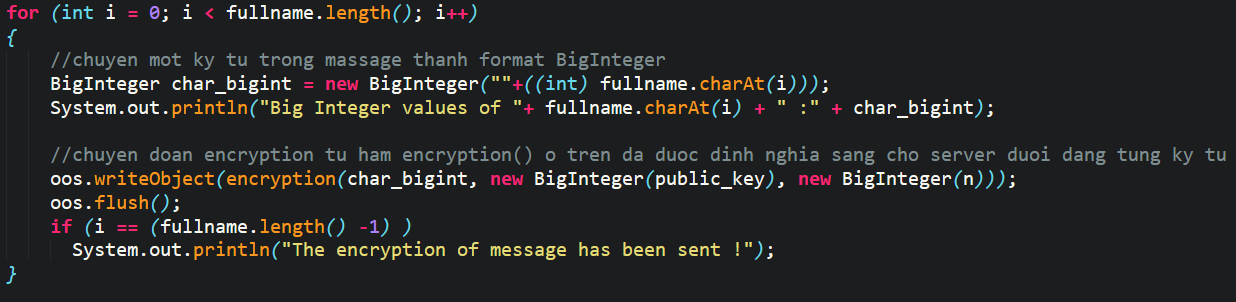


Tiến hành nhập các giá trị public key và n của public key từ server cho client:



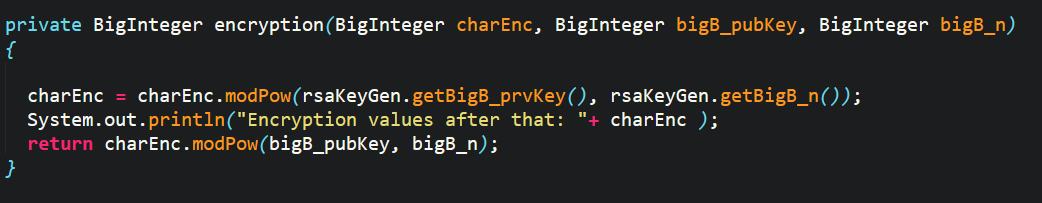
Tiếp theo, ta lần lượt tách các ký tự trong full name ra và dùng for để mã hóa từng ký tự một. quá trình xử lý từng ký tự bao gồm:

ký tự và chuyển sang dạng BigInt được định nghĩa ở bên char\_bigint => sau đó encrypt ngay tại thời điểm tại hàm encryption với ký tự bigint vừa timg được char\_bigint => gửi cho server.



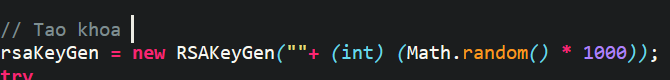
Sau khi đến ký tự cuối ta tiến hành in thông báo đã gửi thông điệp mã hóa để biết đã gửi thông điệp.

Như trên, ta có định nghĩa hàm encryption() thì đây là hàm đó, input sẽ là ký tự BigInterger và khóa public và n và ouput là ký tự sau khi được mã hóa.

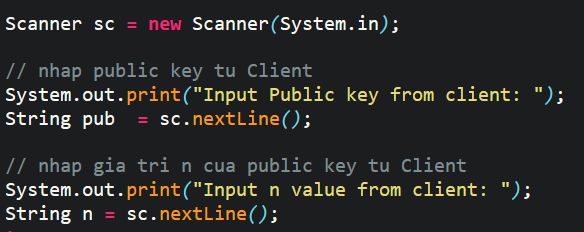


**(4). Server Code and its Description:**

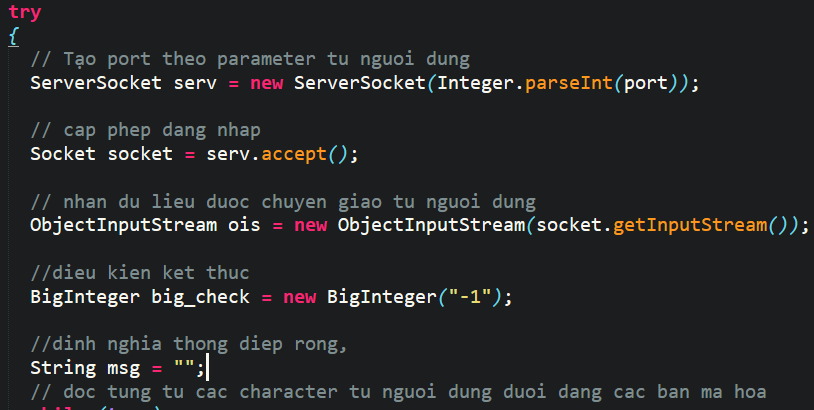
Bước đầu,tạo khóa bang file RSAkeyGen duoc giao



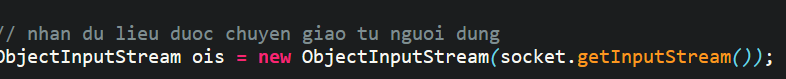
Kế tiếp, ta sẽ tiến hành nhập public key và giá trị n của public vào:



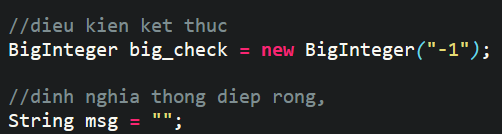
Tạo port local host theo gia tri ma nguoi dung nhap:



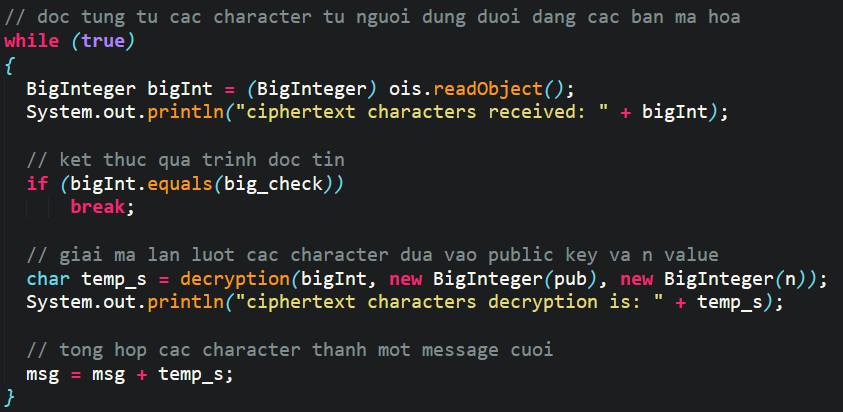
Sau đó, ta gọi hàm giúp nhận các giá trị từ phía client:



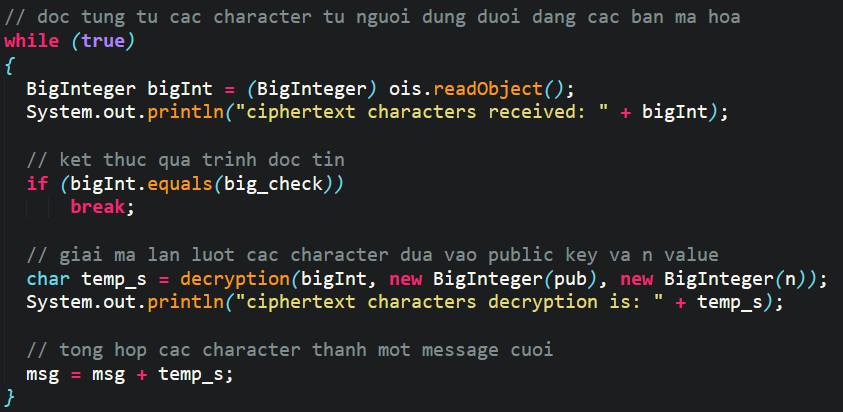
Định nghĩa điều kiện dừng khi người dùng đã kết thúc tin nhắn và thông điệp rỗng để thêm các ký tự mã hóa được vào msg:



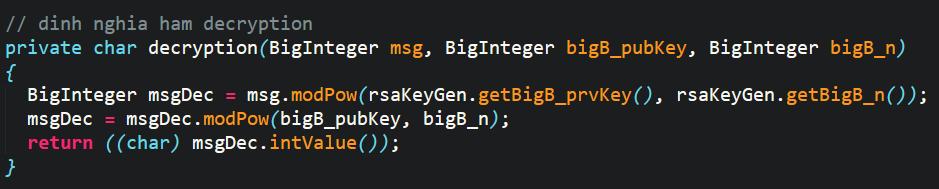
Tiếp đến là quá trình đọc, ở đây ta sẽ định nghĩa ngay cho ký tự đọc vào là dạng BigInterger và in nó ra:



Đến đây ta định nghĩa điều kiện dừng và temp\_s là ký tự sẽ được trả về từ hàm decryption hay là ký tự sau khi được giải mã. Và dòng cuối được dùng để tổng hợp lại đoạn tin nhắn của người dùng.



Hàm decryption() được sử dụng để giải mã ký tự từ public và giá trị n

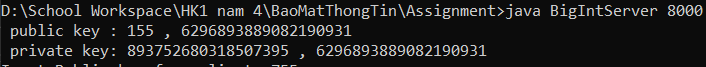


**(5). Execution at the Server Side:**

Đầu tiên ta sẽ compile file BigIntServer.java và RSAKeyGen.java( dùng để khởi tạo khóa).

Sau khi compile ta tiến hành run BigIntServer:

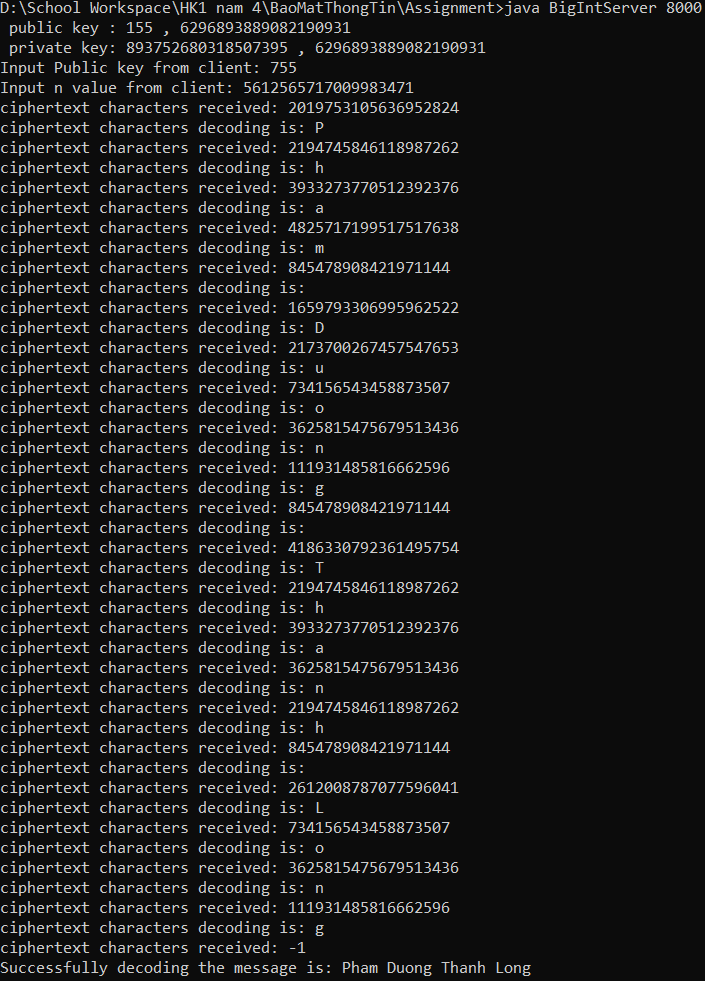
1. Sau khi chạy chương trình sẽ gọi hàm từ RSAKeyGen và tiến hành tạo khóa:



Trong đó:

* **public key**: 155 và giá trị **n**: 6296893889082190931
* **private key**: 893752680318507395 và giá trị **n**: 6296893889082190931 6296893889082190931

1. Sau khi trao đổi public key và giá trị n của public key, server sẽ tiến hành giải mã từng ký tự của cipher text đã nhận:



Sau đó ta tiến hành nối các ký t ự đã nhận được và kết thúc quá trình giải mã:

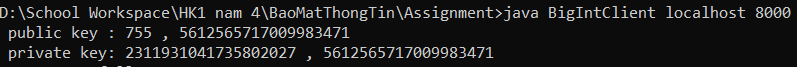


**(6). Execution at the Client Side:**

Đầu tiên ta sẽ compile file BigIntClient.java và RSAKeyGen.java( dùng để khởi tạo khóa). Lúc này bắt buộc BigIntServer phải đang được khởi chạy để có thể tạo liên kết

Sau khi compile ta tiến hành run BigIntClient:

1. Sau khi chạy chương trình sẽ gọi hàm từ RSAKeyGen và tiến hành tạo khóa:



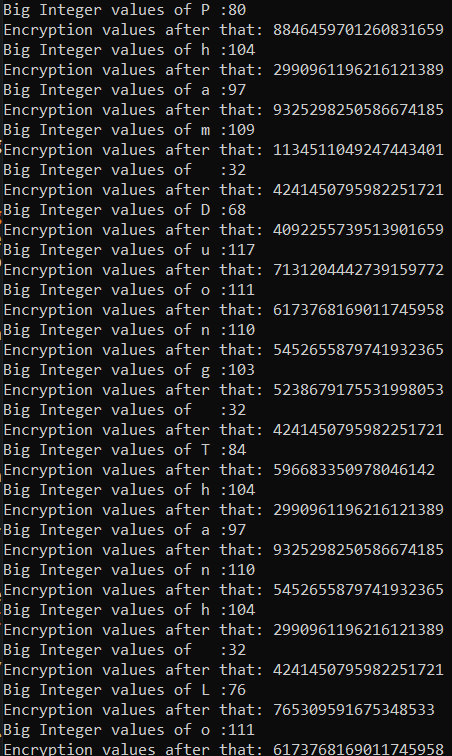
Trong đó:

* **public key**: 755 và giá trị **n**: 5612565717009983471
* **private key**: 2311931041735802027 và giá trị **n**: 5612565717009983471

1. Sau đó ta tiến hành nhập Đoạn mã ASCII mà ở đây là fullname của người dùng ở đây tên mình là: Phạm Dương Thành Long.

* Plain text: : “Phạm Dương Thành Long”

Sau khi nhận được chuỗi ta tiến hành tách ra từng ký tự và chuyển sang dạng BigInt => sau đó chuyển sang encryption ngay tại thời điểm đó trước khi gửi cho server.

…

Thông điệp đã được gửi hoàn tất:



**(7) Tổng kết:**

Phương thức giao tiếp giữa client - server được thực hiện bằng cách sử dụng mô phỏng trên java socket. Hệ mã hóa bất đối xứng RSA cho phép tạo bản mã hóa với private key mà chỉ người tạo mới biết. Khi nhận gói tin, người nhận sẽ dùng public key và giá n của nó được người gửi cung cấp cho riêng người đó cung cấp, Sau đó gói tin băm này được giải mã lần lượt từng ký tự và gọp lại sau đó mang đi so sánh với plain text gốc . Tất nhiêu hệ mã hóa RSA vẫn có những thách thức về an ninh nhất định nhưng dù sao thì nó vẫn khá an toàn.

**Về tính toàn vẹn dữ liệu:**

Tông điệp gửi từ người tạo( client) private key rất khó có thể bị giả mạo. Bởi vì nếu thay đổi tin nhắn thì giá trị hash cũng phải thay đổi theo. Những kẻ nghe lén trong mạng đương nhiên là có thể tìm cách đọc thông điệp gốc và cả hash của nó. Nhưng hắn ta không thể thay đổi tin nhắn được vì hắn không có private key để sửa đổi các giá trị gốc. cho nên tin nhắn là cố định và không thể thay đổi.

Qua việc lập trình java socket như thế này tôi hiểu được nhiều hơn về cơ chế trao đổi của các thành phần trong mạng đặc biệt là quá trình khởi tạo máy chủ và cấp phép lần lượt qua các bước để client lần lượt hưởng đầy đủ các chức năng.

Còn về Big integer cho phép thực hiện các phép toán cộng, trừ, nhân, chia, so sánh, ... với các số Integer không giới hạn miễn sao bạn cấp phát đủ bộ nhớ cho nó, điều này giúp tránh việc hạn chế về việc lưu trữ bất kí một ký tự nào

Còn về giải thuật mã hóa RSA, theo mình về độ an toàn thuật toán này n= p\*q phải là một số đủ lớn, để không có khả năng phân tích nó về mặt tính toán. Để đảm bảo an toàn ta nên chọn các số nguyên tố p và q từ 100 chữ số trở lên. ở đây tôi chọn hẳn giá trị là 1000.