**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**---------------**

****

**BÁO CÁO ĐỒ ÁN**

**LẬP TRÌNH MẠNG MÁY TÍNH**

**Ngành:** Công Nghệ Thông Tin

**Sinh viên thực hiện:**

**Lê Song Vĩ – 1811061712**

**Nguyễn Hữu Minh - 1811062193**

**GVHD : Tống Thanh Văn**

TP. HCM, tháng 01 năm 2022

MỤC LỤC

[CHƯƠNG 1. TỔNG QUAN 3](#_Toc92315099)

[1.1. Mục tiêu đề tài 3](#_Toc92315100)

[1.2. Phương pháp thực hiện 3](#_Toc92315101)

[**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT** 4](#_Toc92315102)

[2.1. Giới thiệu tổng quan 4](#_Toc92315103)

[2.1.1. Khái quát về giao thức Internet 5](#_Toc92315104)

[2.1.2. Giao thức TCP/IP 5](#_Toc92315105)

[2.1.3. Giao thức Internet (IP) 7](#_Toc92315106)

[2.1.4. Tổng quan về UDP 9](#_Toc92315107)

[2.2. Thiết kế và xây dựng chương trình 21](#_Toc92315108)

[2.2.1. Phân tích các chức năng 21](#_Toc92315109)

[2.2.2. Sơ đồ mô tả chương trình 26](#_Toc92315110)

[CHƯƠNG 3. KẾT QUẢ 27](#_Toc92315111)

[3.1. Môi trường triển khai: 27](#_Toc92315112)

[3.2. Kết quả chương trình: 27](#_Toc92315113)

[3.3. Đánh giá và nhận xét 27](#_Toc92315114)

[CHƯƠNG 4. TÀI LIỆU THAM KHẢO 28](#_Toc92315115)

**CHƯƠNG 1. TỔNG QUAN**

**1.1. Mục tiêu đề tài**

Xây chương trình giao diện socket client – server bằng java với giao thức UDP mã hóa và giải mã văn bản với thuật toán mã hóa AES.

Chương trình có thể thực hiện các chức năng sau:

* Phía Client:
* Cho phép nhập văn bản và khóa để mã hóa văn bản trước khi gửi lên server.
* Trao đổi khóa với server
* Nhận kết quả trả về từ client.
* Phía Server:
* Nhận bản mã và khóa sau đó giải mã bản mã để tìm bản rõ.
* Sau khi giải mã xong thì đếm số lượng xuất hiện của các chữ cái trong bản rõ.
* Trả về số lượng xuất hiện của các chữ cái cho client

**1.2. Phương pháp thực hiện**

**-** Tìm kiếm tài liệu liên quan

**-** Tìm hiểu về giao thức mạng UDP

**-** Định hướng chương trình cần làm

**-** Xây dựng chương trình bằng ngôn ngữ Java

**-** Kết luận

**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

**2.1. Giới thiệu tổng quan**

Giao thức Internet (IP) là một giao thức gói không định hướng đã giúp cho hàng triệu máy tính được kết nối với nhau. Từ khi ra đời cách đây 30 năm , IP đã trở thành ngôn ngữ chung cho truyền thông dữ liệu – một giao thức hỗ chợ cho tất cả các ứng dụng. Mức độ liên mạng toàn cầu mà IP đem lại đã cho chúng ta biết về giá trị của sự hợp tác.

Trong phạm vi của những mạng thế hệ mới, IP quan trọng bởi nhiều nguyên nhân. Nó là giao thức hợp nhất giữa máy tính và viễn thông. Nó cung cấp một giải pháp cho việc hợp nhất công nghệ mới trên nguyên tắc kế thừa hạ tầng cơ sở cũ. Nó không mang tính chất độc quyền và luôn là giao thức mở. Nó đưa ra những phương thức hiệu quả và tiết kiệm để hợp nhất thoại và số liệu trên một nền chung, thậm chí nó còn cho phép tạo ra những ứng dụng mới như “không gian dùng chung”. Mặc dù tồn tại nhiều giao thức khác nhưng không giao thức nào có thể sánh được với IP về kinh tế hiệu quả và phạm vi hoạt động toàn cầu của nó.

Khi các ứng dụng mới xuất hiện qua nhiều năm mỗi ứng dụng yêu cầu sự cải tiến tương ứng chất lượng mạng, những sự cải tiến đã tạo điều kiện cho IP thích hợp với những đòi hỏi mới. Thậm chí khi các công nghệ và các giao thức khác được phát triển để đáp ứng những nhu cầu khác nhau như LAN, ATM và chuyển tiếp khung (FR) – những tiến bộ của IP có thể được giữ lại bằng cách chạy phía trên chúng. Có thể nói thực sự về IP rằng nó là giao thức của quá khứ và tương lai.

**2.1.1. Khái quát về giao thức Internet**

Thành công của mạng Internet và Intranet bắt kịp với sự phát triển của kiến trúc mạng máy tính còn gọi là Internet Protocol Suite , được biết đến đó là TCP/IP. Giao thức TCP/IP được sử dụng trong mạng Internet hơn 30 năm qua đã khẳng định rõ ràng khả năng đứng trước những thách thức của truyền thông thế kỷ 21.

**2.1.2. Giao thức TCP/IP**

Cho đến nay TCP/IP là hệ thống giao thức ra đời sớm nhất. Cũng là hệ thống Internet hoàn chỉnh nhất. Để đơn giản hoá việc thiết kế và thực hiện giao thức, thông tin mạng thường căn cứ vào thứ tự lớp khác nhau rồi phân chúng thành những vấn đề con tương ứng. Toàn bộ giao thức cũng phân thành nhiều lớp giao thức khác nhau. Theo ý tưởng này, TCP/IP đã hình thành mô hình phân lớp như sau:

Diagram

Description automatically generated

*Hình 2.1.2.1: Giao thức TCP/IP*

Kiến trúc phân tầng TCP/IP cũng tuân theo mô hình tham chiếu OSI với 4 tầng tuơng ứng như sau:

Diagram

Description automatically generated

*Hình 2.1.2.2: Mô hình OSI và mô hình TCP/IP*

1. Lớp giao diện mạng (Network Interface Physical layer): Hay còn gọi là lớp kết nối, nó tiếp nhận gói dữ liệu IP và phát gói dữ liệu đi theo mạng đã định.
2. Lớp Internet (Network layer): Còn gọi là lớp IP nằm ở lớp thứ 3 trong mô hình OSI, chủ yếu xử lý thông tin giữa các thiết bị. Đối tượng truyền dẫn giữa lớp Internet và lớp giao diện mạng là gói dữ liệu IP (IP Datagram).
3. Lớp truyền dẫn (Transport layer): Nhiệm vụ là cung cấp dịch vụ thông tin giữa các chương trình ứng dụng, đảm bảo số liệu truyền đến đích không có lỗi. Lớp truyền dẫn gói chia làm 2 loại TCP và UDP.
4. Lớp ứng dụng (Application layer): Là lớp cao nhất trong mô hình phân lớp. Thuê bao dùng chương trình ứng dụng để truy nhập mạng Internet TCP/IP và để sử dụng các loại dịch vụ do mạng cung cấp.

**2.1.3. Giao thức Internet (IP)**

IP là giao thức cơ bản để thiết lập mạng TCP/IP . Mục đích của giao thức IP là cung cấp khả năng kết nối các mạng con thành liên mạng truyền dữ liệu . Tất cả các hệ thống thành viên của liên mạng đòi hỏi phải cài đặt IP ở tầng mạng.

IP là giao thức truyền thông kiểu không kết nối (Connectionless) nghĩa là không cần có giai đoạn thiết lập trước khi truyền dữ liệu . Mỗi gói tin truyền đi trên mạng một cách độc lập , chính việc dữ liệu đi tới đích theo nhiều đường khác nhau tạo nên tính mền dẻo cho mạng Internet.

IP là giao thức hoạt động trong lớp 3 của mô hình OSI

Đơn vị thông tin là gói (Packet) , đơn vị dữ liệu truyền trong IP gọi là Datagram

Khuôn dạng của IP (IPv4) như sau:

Timeline

Description automatically generated with low confidence

*Hình 2.1.3.1: Khuôn dạng IP (IPv4)*

Trong RFC 791 đã mô tả chi tiết IPv4 bao gồm các trường sau:

1. Version (Phiên bản): 4 bits mang giá trị là 4, chỉ phiên bản của giao thức IP.
2. IHL (Internet Header Length): 4 bits chỉ thị độ dài mào đầu của Internet, được tính theo đơn vị từ bằng 32 bits.
3. TOS (Type Of Service – Loại hình dịch vụ): 8 bits thiết lập về mức độ ưu tiên và loại hình dịch vụ.
4. Total Length (Tổng độ dài): 16 bits chứa giá trị độ dài tính theo byte của gói IP bao gồm cả Header và Data.
5. Indentification (Nhận Dạng): 16 bits dùng để xác định gói IP hiện thời, sắp xếp lại gói IP theo thứ tự khi gói IP trong chuỗi có gói gửi đi bị hỏng.
6. Flags (cờ): 3 bits được dùng để kiểm soát sự phân đoạn (fragments).
7. Fragment Offset (Mã phân đoạn): 13 bits chỉ vị trí của đơn vị dữ liệu gốc được truyền trong phân đoạn.
8. TTL (Time To Live - Thời gian sống): 8 bits chỉ thời gian tồn tại của đơn vị dữ liệu trong liên mạng trước khi bị coi là mất.
9. Protocol (Giao thức): 8 bits chỉ ra gói tin thuộc loại liên kết nào (UDP sử dụng số 17, TCP sử dụng số 6).
10. Checksum (Tổng kiểm tra): 16 bits mã kiểm soát lỗi phần tiêu đề IP. Các gói bị sai ở giá trị này sẽ bị huỷ bỏ
11. Source Address (Địa chỉ nguồn): 32 bits cung cấp cho phần mềm giao thức IP địa chỉ đích, sử dụng khi phát trở lại.
12. Destanation Address (Địa chỉ đích) : 32 bits dùng cho IP đích kiểm tra số liệu truyền dẫn có chính xác không.
13. Option: Có thể có hoặc không, trường này dùng cho mào đầu lớn hơn 5 từ.
14. Data: Thông thường bao gồm các thông tin số liệu TCP, UDP, ICMP.

**2.1.4. Tổng quan về UDP**

Tiêu chuẩn User Datagram Protocol (UDP) – Giao thức giao vận mạng không kết nối là một giao thức cốt lõi trong bộ giao thức TCP/IP, được sử dụng để đóng gói và gửi dữ liệu ngắn (được gọi là datagram) giữa các máy tính. Giao thức UDP lần đầu được công bố bởi David P.Reed vào năm 1980 và trở thành tiêu chuẩn của do Nhóm chuyên trách kỹ thuật Internet - Internet Engineering Task Force (IETF) vào tháng 8/1980 dưới dạng RFC ([RFC 768](http://www.ietf.org/rfc/rfc768.txt)). Cho đến này, RFC 768 vẫn được khuyến nghị như là đặc tả kỹ thuật chính thức cho giao thức UDP.

Hai giao thức TCP và UDP là các giao thức cốt lõi, nằm tại lớp Truyền tải (Transport) trong bộ giao thức TCP/IP nhằm kết nối các máy tính trên mạng với nhau, chia sẻ và trao đổi dữ liệu. (Bộ giao thức TCP/IP là một bộ các giao thức được phân chia làm 04 tầng: tầng ứng dụng gồm các giao thức như BGP, DNS, FTP, HTTP, SSH…; tầng truyền tải gồm các giao thức như TCP, UDP, SCTP…; tầng mạng internet gồm các giao thức IPv4, IPv6… và tầng truy nhập mạng gồm các giao thức ARP, OSPF, DSL…). Tuy nhiên, giao thức UDP không cung cấp sự tin cậy và thứ tự truyền nhận như giao thức TCP, các gói dữ liệu có thể đến đúng thứ tự hoặc mất mát mà không có sự thông báo. Giao thức UDP nhanh và hiệu quả hơn đối với các ứng dụng truyền tệp tin có kích thước nhỏ và đòi hỏi tuân thủ chặt chẽ về thời gian.

**2.1.4.1. Đặc điểm UDP**

Cổng dịch vụ: UDP sử dụng cổng dịch vụ cho phép giao tiếp, trao đổi dữ liệu giữa các phần mềm/ứng dụng. Cổng dịch vụ sử dụng 16 bít để đánh địa chỉ, do đó, số lượng cổng sẽ từ 0 đến 65.535. Cổng 0 là cổng dự trữ, Cổng 1 đến cổng 1023 được sử dụng cho dịch vụ đã biết, phổ biến (well-known ports), chúng thường được sử dụng để cung cấp các loại dịch vụ mạng như cổng 20 được dùng cho truyền tải dữ liệu [FTP](http://aita.gov.vn/tin-tuc/1353/tieu-chuan-ky-thuat-ve-udcntt-trong-cqnn-tieu-chuan-ftp-giao-thuc-truyen-tep-tin-2), cổng 22 sử dụng cho dịch vụ đăng nhập bảo mật SSH, cổng 520 sử dụng cho giao thức thông tin định tuyến RIP, cổng 992 sử dụng cho giao thức TELNET thông qua TLS/SSL… Cổng 1024 đến cổng 49151 là các cổng được đăng ký, chúng được phân bổ bởi IANA cho các dịch vụ cụ thể. Cổng 49152 tới cổng 65535 là các cổng tự động, không được chỉ định cho một dịch vụ nào cụ thể, có thể sử dụng cho nhiều mục đích khác nhau. Thực tế, các cổng này được dùng chủ yếu khi một máy khách cần liên lạc, kết nối với máy chủ.

Cấu trúc gói tin: UDP là giao thức hướng thông điệp nằm trong tầng truyền tải, được mô tả chi tiết trong RFC 768 của IETF. Phần Tiêu đề trong UDP chứa 04 trường dữ liệu, trong đó 02 trường là trường tùy chọn, gồm trường cổng nguồn và trường kiểm tra tổng.

Graphical user interface, table

Description automatically generated

*Hình 2.1.4.1.1: Cấu trúc gói tin UDP*

Trường Cổng nguồn (tùy chọn): Xác định giá trị cổng của máy gửi thông tin và là cổng trả lời thông điệp (nếu cần thiết). Nếu không sử dụng thì giá trị cổng nên là 0. Nếu máy nguồn là máy khách thì cổng nguồn có thể mang giá trị của cổng tạm thời (49152-65535), nếu máy nguồn là máy chủ thì cổng nguồn mang giá trị cổng phổ biến (0-1023).

Trường Cổng đích: Xác định giá trị cổng của máy nhận. Nếu máy nhận là máy khách thì cổng đích có thể mang giá trị của cổng tạm thời (49152-65535), nếu máy nhận là máy chủ thì cổng đích có thể mang giá trị của cổng phổ biến (0-1023).

Trường Độ dài: Trường có độ dài 16 bít, có chức năng xác định độ dài (theo byte) của phần Tiêu đề và phần Dữ liệu của gói tin UDP. Độ dài tối thiểu của gói tin là 8 byte khi không có dữ liệu và chỉ có phần Tiêu đề. Độ dài tối đa của gói tin là 65535 byte với 8 byte Tiêu đề và 65527 byte dữ liệu.

Trường Kiểm tra tổng (tùy chọn): Trường có độ dài 16 bít sử dụng cho việc kiểm tra lỗi phần Tiêu đề hay phần Dữ liệu.

Giao diện người sử dụng: Giao diện người sử dụng cho phép thiết lập các cổng mới, triển khai trên các cổng này để nhận được dữ liệu, giá trị của cổng nguồn và địa chỉ nguồn. Triển khai việc sử dụng giao thức UDP cho phép một gói tin được gửi đi, xác định dữ liệu, cổng nguồn, cổng đích và địa chỉ mạng máy gửi đi.

Giao diện mạng IP: UDP xác định địa chỉ nguồn và địa chỉ đích, trường giao thức từ phần tiêu đề của gói tin Internet. Giao diện UDP/IP sẽ trả lại toàn bộ gói tin Internet bao gồm phần tiêu đề của gói tin Internet để thực hiện bóc tách thông tin. Giao diện UDP/IP cho phép gói tin UDP được đóng gói trong tất cả gói tin Internet với địa chỉ đích phù hợp với giao thức IP. Giao thức IP sẽ kiểm tra tính thống nhất của các trường trong phần tiêu đề của gói tin Internet.

Kỹ thuật chống tắc nghẽn: Do thiếu cơ chế kiểm soát tắc nghẽn, kiểm soát luồng, việc sử dụng UDP dễ dẫn đến hiệu ứng tắc nghẽn dây chuyền, hay nói cách khác máy gửi gói tin UDP không kiểm soát, phát hiện được quá trình tắc nghẽn. Các kỹ thuật như sử dụng bộ định tuyến sử dụng hàng đợi hoặc kỹ thuật bỏ gói tin là những công cụ hữu hiệu nhằm giảm tải khi sử dụng giao thức UDP. Giao thức kiểm soát tác nghẽn gói tin (DCCP) được thiết kế như một giải pháp cho vấn đề bằng cách thêm hành vi kiểm soát tắc nghẽn cho thiết bị đầu cuối cho các dòng dữ liệu UDP như dòng thông tin đa phương tiện (streaming media).

**2.1.5. Giới thiệu về hệ mã khối**

Trong mật mã học, mã hóa khối là những thuật toán mã hóa đối xứng hoạt động trên những khối thông tin có độ dài xác định (block) với những chuyển đổi xác định. Chẳng hạn một thuật toán mã hóa khối có thể xử lý khối 128 byte đầu vào và biến nó thành khối 128 byte ở đầu ra. Quá trình chuyển đổi còn sử dụng thêm một tham số nữa: khóa bí mật để cá biệt hóa quá trình. Việc giải mã cũng diễn ra tương tự: xử lý khối mã hóa 128 byte cùng với khóa để trả về khối 128 byte bản rõ ban đầu.

Để mã hóa những văn bản có độ dài vượt quá độ dài của khối, người ta sử dụng thuật toán theo một chế độ mã hóa khối nào đó.

Phân biệt với mã hóa khối là mã hóa dòng. Mã hóa dòng làm việc trên từng bít của dòng dữ liệu và quá trình biến đổi thay đổi theo quá trình mã hóa. Tuy nhiên, sự phân biệt giữa 2 phương pháp nhiều khi không rõ ràng vì mã hóa khối khi hoạt động theo một chế độ nào đó thì có tác dụng như một phương pháp mã hóa dòng.

Thuật toán mã hóa khối ra đời sớm và có nhiều ảnh hưởng là thuật toán DES (Data Encryption Standard - Tiêu chuẩn mã hóa dữ liệu) do công ty IBM phát triển và được ban hành làm tiêu chuẩn năm 1977. Tiêu chuẩn thay thế DES có tên là AES (Advanced Encryption Standard - Tiêu chuẩn mã hóa nâng cao) được ban hành năm 2001.

**2.1.6. Giới thiệu về hệ mã khối AES**

AES là viết tắt của Advanced Encryption Standard, chuẩn mã hóa dữ liệu rất phổ biến, dùng cho nhiều mục đích và được cả chính phủ Mỹ sử dụng để bảo vệ các dữ liệu tuyệt mật.

AES là kiểu mã hóa đối xứng dạng khối, nghĩa là mỗi khối văn bản có một kích thước nhất định (128 bit) được mã hóa, khác với mã hóa dạng chuỗi khi từng kí tự được mã hóa. Đối xứng nghĩa là khóa để mã hóa và giải mã đều là một.

**2.1.6.1. Yêu cầu của AES**

Phương pháp mã hóa theo khối có kích thước khối dữ liệu đầu vào và đầu ra là 128 bit, độ dài khóa có thể thay đổi linh hoạt với các giá trị 128, 192 hay 256 bit. Phương pháp mã hóa này thích hợp ứng dụng trên nhiều hệ thống khác nhau, từ các thẻ thông minh cho đến các máy tính cá nhân.

Chuẩn mã mới mạnh và nhanh hơn Triple DES. Mã mới có cơ sở lý thuyết mạnh để thời gian sống của chuẩn khoảng 20 - 30 năm (cộng thêm thời gian lưu trữ).

Khi đưa ra thành phần yêu cầu cung cấp chi tiết thiết kế và đặc tả đầy đủ. Đảm bảo rằng chuẩn mã mới cài đặt hiệu quả trên cả C và Java.

**2.1.6.2. Cơ sở toán học của AES**

Trong AES các phép toán cộng và nhân được thực hiện trên các byte trong trường hữu hạn GF(28)

* *Phép cộng:*

A = (a1 a2 a3 a4 a5 a6 a7 a8); B = (b1 b2 b3 b4 b5 b6 b7 b8); C = A + B = (c1 c2 c3 c4 c5 c6 c7 c8), trong đó: Ci = ai + bi.

Ví dụ 2.5:

A = 56H; B = 3DH

Dạng cơ số Hecxa: 56H + 3DH = 93

Dạng nhị phân: 01010110 + 00111101 = 10010011

Dạng đa thức: (x6 + x4 + x2 + x) + (x5 + x4 + x3 + x2 + 1) = (x7 + x4 + x + 1)

* *Phép nhân:*

A = (a1 a2 a3 a4 a5 a6 a7 a8); B = (b1 b2 b3 b4 b5 b6 b7 b8); C = A.B = (c1 c2 c3 c4 c5 c6 c7 c8)

Ví dụ 2.6:

A = C3H; B = 85H

Dạng cơ số Hecxa: (C3H).(85H) = AEH

Dạng nhị phân: (11000011).(10000101) = 10101110

Dạng đa thức: (x7 + x6 + x + 1).(x7 + x2 + 1) = (x7 + x5 + x3 + x2 + 1)

**2.1.6.3. Chu trình tạo khóa con AES**

AES thực hiện việc mở rộng khóa dựa trên khóa gốc K, tạo thành chu trình tạo khóa để sinh ra 10, 12 hoặc 14 khóa con, tương ứng với 10, 12 hoặc 14 chu kỳ lặp của giải thuật AES.

Việc mở rộng khóa chính tạo thành bảng khóa mở rộng. Bảng khóa mở rộng là mảng 1 chiều chứa các từ, mỗi từ có độ dài 4 byte, được ký hiệu W[Nb\*(Nr+1)] (với Nb = 4). Việc phát sinh bảng khóa mở rộng phụ thuộc vào độ dài Nk của khóa chính.

Chu trình tạo khóa con AES sử dụng hai hàm:

* SubWord() thực thiện việc thay thế từng byte thành phần của từ 4 byte được đưa vào và trả về kết quả là một từ 4 byte đã được thay thế. Việc thay thế này sử dụng bảng thay thế S-box.
* RotWord() thực hiện việc dịch chuyển xoay vòng 4 byte thành phần (a, b, c,d) của từ được đưa vào. Kết quả trả về của hàm RotWord là một từ 4 byte đã được dịch chuyển (b, c, d, a).

Các hằng số chu kỳ Rcon[i] được xác định:

* Rcon[i] = [xi-1, {00}, {00}, {00}]
* Trong đó: xi-1  {02}i-1 trong trường GF(28)

Như vậy ta có bảng hằng số mở rộng với trường hợp Nr = 10 như sau:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 | 1B | 36 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

*Bảng 2.1.6.3.1: Bảng hằng số mở rộng Rcon của AES – 128*

Khóa của chu kỳ thứ i bao gồm các từ 4 byte có chỉ số từ Nb\*i đến Nb\*(i+1) – 1 (với Nb = 4) của bảng mã khóa mở rộng. Như vậy mã khóa của chu kỳ thứ i bao gồm các phần tử từ W[Nb\*i], W[Nb\*i + 1],…W[Nb\*(i+1) – 1],

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| W0 | W1 | W2 | W3 | W4 | W5 | W6 | W7 | W8 | W9 | W10 | W11 | … |

|  |  |  |  |
| --- | --- | --- | --- |
| Khóa con chu kỳ 0 | Khóa con chu kỳ 1 | Khóa con chu kỳ 2 | … |

*Bảng 2.1.6.3.2: Bảng khóa mở rộng AES - 128*

**2.1.6.4. Quá trình mã hóa AES**

Giải thuật AES bao gồm nhiều bước biến đổi được thực hiện tuần tự, kết quả đầu ra của bước biến đổi này sẽ là đầu vào của bước biến đổi kia. Kết quả trung gian giữa các bước biến đổi được gọi là trạng thái (State). Độ dài của khối đầu vào, khối đầu ra cũng như độ dài của khối trung gian State là 128 bit. Được biểu diễn bằng một ma trận gồm 4 dòng và 4 cột.

Độ dài của khóa K trong giải thuật AES có thể là 128, 192 hoặc 256 bit. Khóa được biểu diễn bằng một ma trận gồm 4 dòng và Nk cột (Nk = 4, 6 hoặc 8; Nk được tính bằng độ dài của khóa chia 32). Số lượng chu kỳ tính toán trong giải thuật AES được ký hiệu là Nr, độ lớn của Nr phụ thuộc vào độ dài của khóa. Nr được xác định theo công thức:

Nr = max{Nb, Nk} + 6

|  |  |  |  |
| --- | --- | --- | --- |
|  | Độ dài khóa  (Nk words) | Kích thước khối  (Nb words) | Số chu kỳ  (Nr) |
| AES – 128 | 4 | 4 | 10 |
| AES – 192 | 6 | 4 | 12 |
| AES – 256 | 8 | 4 | 14 |

*Bảng 2.1.6.4.1: Mối liên hệ giữa Nk, Nb và Nr*

Quá trình mã hóa của giải thuật AES trải qua 10, 12 hoặc 14 chu kỳ, tương ứng với độ dài của khóa là 128, 192 hoặc 256 bit. Mỗi chu kỳ bao gồm 4 bước được thự hiện tuần tự:

Bước 1: AddRoundKey - Mỗi cột của trạng thái đầu tiên lần lượt được kết hợp với một khóa con theo thứ tự từ đầu dãy khóa.

Các khóa con này được tạo ra từ quá trình tạo khóa con.

A picture containing text, crossword puzzle

Description automatically generated

*Hình 2.1.6.4.1: AddRoundKey*

Bước 2: SubBytes - đây là phép thế (phi tuyến) trong đó mỗi byte trong trạng thái sẽ được thế bằng một byte khác theo bảng tra (Rijndael S-box).Diagram

Description automatically generated

*Hình 2.1.6.4.2: SubBytes*

Bước 3: ShiftRows - dịch chuyển, các hàng trong trạng thái được dịch vòng theo số bước khác nhau.

Table

Description automatically generated

*Hình 2.1.6.4.3: ShiftRows*

Bước 4: MixColumns - quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.

A picture containing text, crossword puzzle, black

Description automatically generated

*Hình 2.1.6.4.4: MixColumns*

Các bước của quá trình mã hóa được thực hiện trên trạng thái hiện hành S. Kết quả S’ của mỗi bước sẽ trở thành đầu vào của bước tiếp theo.

Diagram

Description automatically generated

*Hình 2.1.6.4.5: Quá trình Key Schedule*

Quá trình giải mã là một quá trình ngược của quá trình mã hóa AES. Quá trình giải mã cũng trải qua 10, 12 hoặc 14 chu kỳ tương ứng với số chu kỳ của quá trình mã hóa.

Mỗi chu kỳ gồm 4 bước được thực hiện tuần tự với nhau gồm InvSubBytes, InvShiftRows, InvMixColumns (là các phép biến đổi ngược với SubBytes, ShiftRows, MixColumns) và bước AddRoundKey.

Quá trình giải mã được thể hiện như lưu đồ dưới đây

* Khối dữ liệu đầu vào là Ciphertext được sao chép vào mảng trạng thái S.
* Thực hiện thao tác AddRoundKey đầu tiên trước khi thực hiện các chu kỳ mã hóa. Sử dụng khóa ở chu kỳ thứ Nr của chu trình mã hóa.
* Mảng trạng thái sau đó sẽ trải qua Nr = 10, 12 hay 14 chu kỳ biến đổi (tương ứng với 10, 12 hay 14 chu kỳ mã hóa).
* Nr – 1 chu kỳ đầu tiên: mỗi chu kỳ gồm 4 bước biến đổi liên tiếp nhau.
* Chu kỳ thứ Nr, thao tác InvMixColumns được thay thế bằng thao tác AddRoundkey.

Diagram

Description automatically generated

*Hình 2.1.6.4.6: Quy trình giải mã AES*

**2.1.7. Mô hình Client – Server**

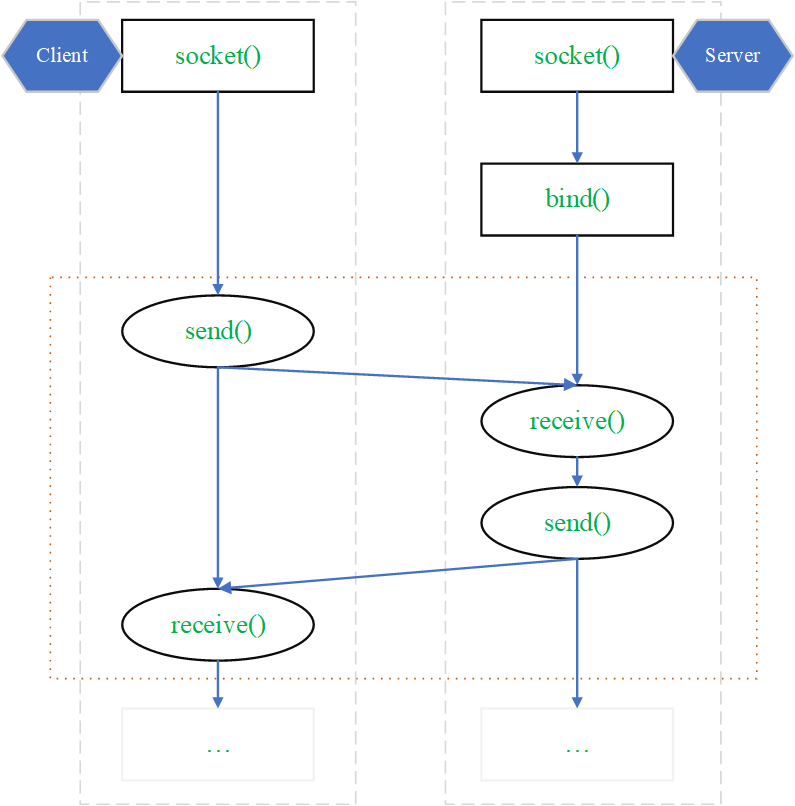
Mô hình Client – Server là một mô hình nổi tiếng trong mạng máy tính, được áp dụng rất rộng rãi và là mô hình của mọi trang web hiện có. Ý tưởng của mô hình này là máy con (đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.

**2.1.7.1. Quá trình trao đổi dữ liệu của mô hình Client – Server**

Quá trình trao đổi dữ liệu gồm các bước:

1. Truyền một yêu cầu từ tiến trình Client tới tiến trình Server
2. Yêu cầu được Server xử lý
3. Truyền đáp ứng cho Client

Mô hình Client/Server thường được cài đặt dựa trên các thao tác cơ bản là gửi (send) và nhận (receive).



Hình 2.1.7.1.1: Sơ đồ mô tả quá trình trao đổi dữ liệu giữa Client – Server

**2.2. Thiết kế và xây dựng chương trình**

**2.2.1. Phân tích các chức năng**

Xây chương trình giao diện socket client – server bằng java với giao thức UDP mã hóa và giải mã văn bản với thuật toán mã hóa AES. Chương trình có thể thực hiện các chức năng sau:

Ví AES là khóa đối xứng, mã hóa và giải mã chỉ sử dụng 1 khóa bí mật duy nhất. Vậy nên nhóm em sử dụng thêm thuật toán mã hóa RSA để thực hiện quá trình trao đổi khóa bí mật của AES, tránh việc bị rò rỉ khóa bí mật AES giữa Client – Server. Chương trình sẽ thực hiện các bước như sau:

* *Bước 1*: Nhận văn bản cùng mật khẩu mà người dùng nhập vào từ Client

File *ClientFrm.java*

ClientThread wfr **=** **new** ClientThread**();**

wfr**.**setMainText**(this.**txtInput**.**getText**());**

wfr**.**setAesKey**(this.**txtPassword**.**getText**());**

wfr**.**start**();**

* *Bước 2*: Client tạo cặp khóa RSA và gửi khóa công khai tới Server

File *ClientThread.java*

KeyPairGenerator keyPairGenerator **=** KeyPairGenerator**.**getInstance**(**"RSA"**);**

keyPairGenerator**.**initialize**(**keysize**);**

KeyPair keyPair **=** keyPairGenerator**.**generateKeyPair**();**

publicKey **=** **(**RSAPublicKey**)** keyPair**.**getPublic**();**

privateKey **=** **(**RSAPrivateKey**)** keyPair**.**getPrivate**();**

send**(**String**.**valueOf**(**publicKey**),** ipServer**,** PORT**);**

* *Bước 3*: Server nhận được khóa RSA công khai của Client, Server tạo cặp khóa RSA và gửi khóa công khai tới Client

File *ServerThread.java*

packet **=** receive**();**

host **=** packet**.**getAddress**();**

port **=** packet**.**getPort**();**

message **=** **new** String**(**packet**.**getData**()).**trim**();**

**if** **(**message **==** **null)** **return;**

Matcher m **=** helper**.**InfoExtract**(**message**);**

**if(!**m**.**find**())** **{**

System**.**err**.**println**(**"Dữ liệu không hợp lệ!"**);**

**return;**

**}**

String ClientModulus **=** m**.**group**(**1**).**trim**();**

String ClientPublicExponent **=** m**.**group**(**2**).**trim**();**

BigInteger bigInteger **=** **new** BigInteger**(**ClientModulus**);**

BigInteger bigInteger1 **=** **new** BigInteger**(**ClientPublicExponent**);**

RSAPublicKeySpec keyspec **=** **new** RSAPublicKeySpec**(**bigInteger**,** bigInteger1**);**

KeyFactory kf **=** KeyFactory**.**getInstance**(**"RSA"**);**

publicKeyofClient **=** **(**RSAPublicKey**)** kf**.**generatePublic**(**keyspec**);**

**...**

send**(**String**.**valueOf**(**publicKey**),** host**,** port**);**

* *Bước 4*: Client nhận được khóa RSA công khai của Server. Client tạo khóa AES mới dựa trên chuỗi mật khẩu với độ dài tùy ý mà người dùng nhập vào, đưa nó vào quá trình Message Digest để cho ra khóa AES với độ dài 16 bytes dựa trên chuỗi mật khẩu ban đầu của người dùng. Khóa này sau đó được mã hóa bằng thuật toán RSA với khóa công khai của Server. Client gửi gói dữ liệu mã hóa này tới Server.

File *ClientThread.java*

DatagramPacket packet **=** receive**();**

String txtContent **=** **new** String**(**packet**.**getData**()).**trim**();**

Matcher m **=** **this.**helper**.**InfoExtract**(**txtContent**);**

**if(!**m**.**find**())** **{**

System**.**err**.**println**(**"Dữ liệu không hợp lệ!"**);**

**return;**

**}**

String serverModulus **=** m**.**group**(**1**).**trim**();**

String serverExponent **=** m**.**group**(**2**).**trim**();**

BigInteger modulos **=** **new** BigInteger**(**serverModulus**);**

BigInteger exponent **=** **new** BigInteger**(**serverExponent**);**

RSAPublicKeySpec rsaPublicKeySpec **=** **new** RSAPublicKeySpec**(**modulos**,** exponent**);**

keyFactory **=** KeyFactory**.**getInstance**(**"RSA"**);**

publicKeyOfServer **=** **(**RSAPublicKey**)** keyFactory**.**generatePublic**(**rsaPublicKeySpec**);**

**...**

KeyGenerator kg **=** KeyGenerator**.**getInstance**(**"AES"**);**

kg**.**init**(**128**);**

ClientSecretKey **=** **this.**helper**.**convertKey**(this.**aesKeyInput**);**

Cipher c **=** Cipher**.**getInstance**(**"RSA"**);**

**assert** c **!=** **null;**

c**.**init**(**Cipher**.**ENCRYPT\_MODE**,** publicKeyOfServer**);**

byte**[]** encodedBytes **=** c**.**doFinal**(**ClientSecretKey**.**getEncoded**());**

**assert** encodedBytes **!=** **null;**

sendByte**(**encodedBytes**,** ipServer**,** PORT**);**

* *Bước 5*: Server nhận được gói dữ liệu từ Client, tiến hành giải mã RSA bằng khóa bí mật của Server để nhận được khóa bí mật AES. Sau đó gửi một lệnh ping tới Client để yêu cầu nhận dữ liệu văn bản để xử lý.

File *ServerThread.java*

packet **=** receive**();**

byte**[]** EncryptedClientAES **=** packet**.**getData**();**

EncryptedClientAES **=** Arrays**.**copyOf**(**EncryptedClientAES**,** 256**);**

Cipher cipher **=** Cipher**.**getInstance**(**"RSA"**);**

cipher**.**init**(**Cipher**.**DECRYPT\_MODE**,** privateKey**);**

byte**[]** decryptedAESKeyOfClient **=** cipher**.**doFinal**(**EncryptedClientAES**);**

clientAESKey **=** **(**SecretKey**)** **new** SecretKeySpec**(**decryptedAESKeyOfClient**,** 0**,** decryptedAESKeyOfClient**.**length**,** "AES"**);**

**...**

send **(**"ping"**,** host**,** port**);**

* *Bước 6*: Client nhận được một lệnh ping từ Server, Client bắt đầu mã hóa văn bản người dùng nhập vào bằng thuật toán AES sau đó gửi tới Server.

File *ClientThread.java*

receive**();**

send**(this.**encrypt**(this.**mainText**),** ipServer**,** PORT**);**

* *Bước 7*: Server nhận được dữ liệu mã hóa từ Client, tiến hành giải mã và tính toán, xử lý văn bản. Sau đó gửi kết quả đã hoàn thành về cho Client.

File *ServerThread.java*

packet **=** receive**();**

message **=** **new** String **(**packet**.**getData**()).**trim**();**

**if** **(**message**.**trim**().**length**()** **==** 0**)** **{**

System**.**err**.**println**(**"Dữ liệu không hợp lệ"**);**

**return;**

**}**

TextPack txtPack **=** **new** TextPack**(this.**helper**.**charCount**(this.**decrypt**(**message**)));**

**try** **(**ByteArrayOutputStream bos **=** **new** ByteArrayOutputStream**();**

ObjectOutputStream out **=** **new** ObjectOutputStream**(**bos**))** **{**

out**.**writeObject**(**txtPack**);**

out**.**flush**();**

byte**[]** yBytes **=** bos**.**toByteArray**();**

send**(this.**encrypt**(**Base64**.**getEncoder**().**encodeToString**(**yBytes**)),** host**,** port**);**

**}**

* *Bước 8*: Client nhận được kết quả tính toán từ Server và hiển thị nó lên cho người dùng.

File *ClientThread.java*

DatagramPacket packet **=** receive**();**

String fromServerStr **=** **this.**decrypt**(new** String**(**packet**.**getData**()).**trim**());**

**try** **(**ByteArrayInputStream bis **=** **new** ByteArrayInputStream**(**Base64**.**getDecoder**().**decode**(**fromServerStr**));**

ObjectInputStream in **=** **new** ObjectInputStream**(**bis**);)**

**{**

TextPack txtPack **=** **(**TextPack**)**in**.**readObject**();**

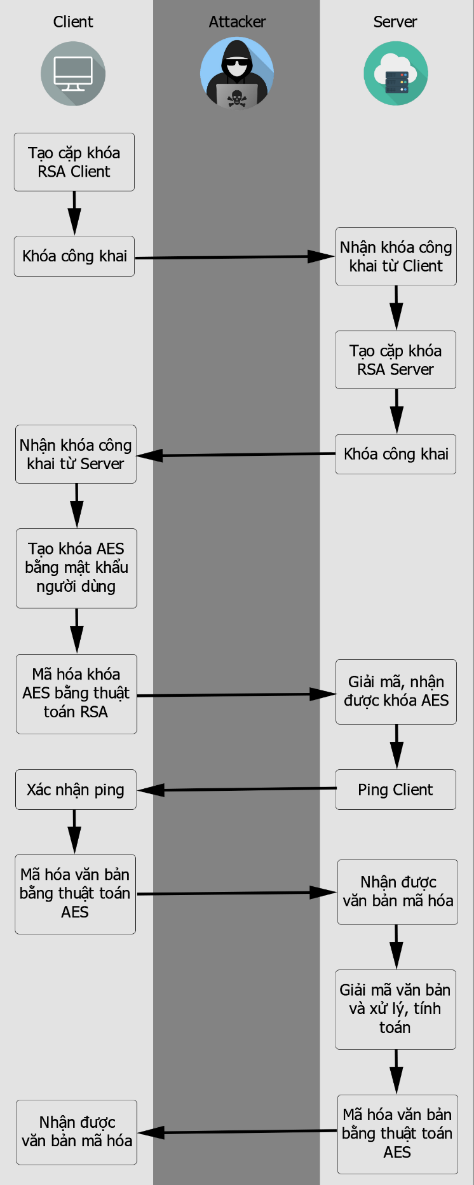
resultContent **=** **this.**helper**.**txtPackToText**(**txtPack**,** **true);**

**}** **catch** **(**ClassNotFoundException ex**)** **{**

Logger**.**getLogger**(**ClientThread**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

**}**

**2.2.2. Sơ đồ mô tả chương trình**



*Hình 2.2.2.1. Sơ đồ mô tả hoạt động của chương trình*

**CHƯƠNG 3. KẾT QUẢ**

**3.1. Môi trường triển khai:**

Chương trình được chạy trên NetBeans IDE trên hệ điều hành Windows.

**3.2. Kết quả chương trình:**

Graphical user interface, application

Description automatically generated

*Hình 3.2.1: Form demo của chương trình*

**3.3. Đánh giá và nhận xét**

* Chương trình còn đơn giản, chưa có chiều sâu.
* Về hướng phát triển, em sẽ tích hợp tính năng trao đổi khóa AES của phần mềm này vào một phần mềm khác mà em phát triển trong tương lai.

**CHƯƠNG 4. TÀI LIỆU THAM KHẢO**

* Giáo trình Lập trình mạng máy tính, Trường ĐH Công Nghệ TP. HCM
* Tài liệu Java (Oracle): <https://docs.oracle.com/en/java/>