**TRƯỜNG ĐẠI HỌC KINH DOANH VÀ CÔNG NGHỆ HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo

Description automatically generated**

**ĐỒ ÁN ĐIỆN TOÁN DI ĐỘNG**

**Xây dựng phần mềm đồng bộ**

**bảng nhớ tạm ClipSync**

**Sinh viên: Mai Quang Hải**

**Mã sinh viên: 2823220209**

**Lớp: TH28.27**

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 3](#_Toc196298984)

[**CHƯƠNG 1. TỔNG QUAN** 4](#_Toc196298985)

[***1.1. Giới thiệu đề tài*** 4](#_Toc196298986)

[***1.2. Các kiến thức sử dụng trong đề tài*** 4](#_Toc196298987)

[1.2.1. Phát triển phần mềm cho smartphone Android 4](#_Toc196298988)

[1.2.2. Phát triển phần mềm VB.NET cho Windows 4](#_Toc196298989)

[1.2.4. Giao tiếp mạng 5](#_Toc196298990)

[***1.3. Các công cụ sử dụng trong đề tài*** 5](#_Toc196298991)

[**CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ** 6](#_Toc196298992)

[***2.1. Phân tích và thiết kế hệ thống*** 6](#_Toc196298993)

[2.1.1. Phân tích chương trình 6](#_Toc196298994)

[2.1.2. Phân tích thiết kế cơ chế giao tiếp mạng 6](#_Toc196298995)

[2.1.2.1. Giao thức TCP 6](#_Toc196298996)

[2.1.2.2. Giao thức UDP 7](#_Toc196298997)

[2.1.2.3. Những sự khác nhau giữa hai giao thức TCP và UDP 7](#_Toc196298998)

[2.1.2.4. Phân tích cơ chế giao tiếp mạng cho ứng dụng ClipSync 8](#_Toc196298999)

[2.1.3. Thiết kế các gói tin 11](#_Toc196299000)

[2.1.3.1. Gói tin broadcast 11](#_Toc196299001)

[2.1.3.2. Gói tin TCP 11](#_Toc196299002)

[2.1.4. Phân tích thiết kế cơ chế xác thực OTP 12](#_Toc196299003)

[2.1.5. Phân tích thiết kế cơ chế mã hoá dữ liệu 13](#_Toc196299004)

[2.1.5.1. Thuật toán mã hoá đối xứng 14](#_Toc196299005)

[2.1.5.2. Thuật toán mã hoá bất đối xứng 14](#_Toc196299006)

[2.1.5.3. Cơ chế mã hoá dữ liệu cho phần mềm ClipSync 14](#_Toc196299007)

[**CHƯƠNG 3. CHƯƠNG TRÌNH PHẦN MỀM** 21](#_Toc196299008)

[***3.1. Giao diện*** 21](#_Toc196299009)

[3.1.1. Giao diện phía Windows 21](#_Toc196299010)

[3.1.2. Giao diện phía Android 23](#_Toc196299011)

[***3.2. Một số đoạn code chính*** 24](#_Toc196299012)

[***3.3. Những hạn chế của chương trình*** 24](#_Toc196299013)

[**TÀI LIỆU THAM KHẢO** 25](#_Toc196299014)

# **LỜI NÓI ĐẦU**

Thế giới ngày nay đã có nhiều phát triển mạnh mẽ về công nghệ thông tin (CNTT). Từ một tiềm năng thông tin đang trở thành một tài nguyên thực sự, trở thành một hàng hóa trong xã hội, góp phần tạo ra sự thay đổi to lớn trong lực lượng sản xuất, cơ sở hạ tầng, cấu trúc kinh tế, tính chất lao động và cả cách thức quản lí trong các lĩnh vực của xã hội. Trong những năm gần đây nền CNTT nước ta đã có nhiều phát triển trong mọi lĩnh vực của cuộc sống cũng như trong lĩnh vực quản lí xã hội. Một trong những lĩnh vực mà máy tính được sử dụng nhiều nhất là hệ thống thông tin quản lý. Sự phát triển của Internet đã đưa con người vào kỷ nguyên xa lộ thông tin. Thông qua mạng Internet, con người có thể mua bán hàng hóa, trao đổi thông tin rộng rãi trên toàn cầu. Công nghệ đã thúc đẩy mạnh mẽ việc khai thác thông tin trên mạng với tính năng linh hoạt và dễ sử dụng.

Cùng với sự phát triển của ngành công nghệ thông tin, máy tính càng trở nên gần gũi và thông dụng với cuộc sống của chúng ta. Máy tính có khả năng hỗ trợ hiệu quả những công việc khó khăn, phức tạp trong mọi lĩnh vực của cuộc sống, đặc biệt là trong công tác quản lý tại các doanh nghiệp và cơ quan, như bảo mật thông tin, xem và chỉnh sửa dữ liệu một cách dễ dàng, nhanh chóng và hiệu quả. Nhiều phần mềm đã được phát triển nhằm đáp ứng yêu cầu phức tạp trong các lĩnh vực của cuộc sống, đặc biệt là trong công tác quản lý tại các doanh nghiệp và cơ quan.

Trong suốt quá trình học tập, em đã được các thầy cô giáo cung cấp và truyền đạt những kiến thức cần thiết trong lĩnh vực công nghệ thông tin. Ngoài ra em còn được rèn luyện tinh thần học tập, làm việc độc lập sáng tạo. Trong khuôn khổ một đề án môn học dưới sự hướng dẫn tận tình của thầy cô và sự tích lũy kiến thức của bản thân, em đã xây dựng đề tài “Xây dựng phần mềm đồng bộ bảng nhớ tạm ClipSync”.

# **CHƯƠNG 1. TỔNG QUAN**

## ***1.1. Giới thiệu đề tài***

Phần mềm đồng bộ bảng nhớ tạm ClipSync là phần mềm giúp người sử dụng có thể truyển tải dữ liệu bảng nhớ tạm giữa các thiết bị một cách tức thì thông qua hệ thống mạng nội bộ (LAN). Phần mềm ClipSync được phát triển trong đề tài này sử dụng mô hình giao tiếp ngang hàng (P2P), bao gồm hai hoặc nhiều máy của người dùng tạo thành một mạng.

## ***1.2. Các kiến thức sử dụng trong đề tài***

### 1.2.1. Phát triển phần mềm cho smartphone Android

* Phát triển ứng dụng Android với Java
* Sử dụng SharedPreferences để lưu trữ thông tin nội bộ của ứng dụng
* Hiểu và xử lý chính sách bảo mật trên các phiên bản Android 10 (API 29) trở lên về vấn đề truy xuất dữ liệu Clipboard
* Sử dụng cơ bản hệ thống build Gradle
* Sử dụng dịch vụ Accessibility Service
* Làm việc với giao diện LinearLayout
* Xử lý đa tiến trình

### 1.2.2. Phát triển phần mềm VB.NET cho Windows

* Phát triển ứng dụng Windows với VB.NET

1.2.3. Bảo mật dữ liệu người dùng

* Sử dụng hàm băm SHA512 băm mật khẩu người dùng trước khi lưu vào SharedPreferences
* Mã hoá dữ liệu bằng thuật toán XChaCha-Poly1305 trước khi truyền tải qua mạng
* Xác thực giữa các máy bằng thuật toán TOTP sinh mã OTP

### 1.2.4. Giao tiếp mạng

* Gửi gói UDP Broadcast để ra tín hiệu đồng bộ
* Xác thực OTP và truyền tải dữ liệu qua TCP Socket

## ***1.3. Các công cụ sử dụng trong đề tài***

Trong quá trình phát triển ứng dụng, việc lựa chọn và sử dụng các công cụ phù hợp đóng vai trò quan trọng trong việc đảm bảo hiệu suất, tính bảo mật và khả năng mở rộng của hệ thống. Các công cụ này không chỉ hỗ trợ lập trình và kiểm thử mà còn giúp quản lý mã nguồn, phân tích lỗi và tối ưu hóa quy trình phát triển. Dưới đây là các công cụ chính đã được sử dụng trong đề tài để triển khai ứng dụng trên cả nền tảng Android và Windows.

* **Android Studio:** IDE dùng để phát triển ứng dụng cho Android
* **Visual Studio 2022:** IDE dùng để phát triển ứng dụng bằng VB.NET cho Windows
* **Wireshark:** Công cụ dùng để bắt và phân tích các gói mạng
* **Nmap:** Công cụ dùng để quét mạng
* **Ettercap:** Bộ công cụ giúp thực hiện các hoạt động tấn công Man in the Middle (MITM) để giám sát lưu lượng giữa các client cũng như dùng để tấn công kiểm thử mức độ bảo mật của phần mềm.

# **CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ**

## ***2.1. Phân tích và thiết kế hệ thống***

### 2.1.1. Phân tích chương trình

Phần mềm đồng bộ bảng nhớ tạm ClipSync gồm các chức năng như sau:

* Đọc ghi dữ liệu Clipboard
* Mã hoá và giải mã dữ liệu
* Chức năng xác thực bằng OTP
* Giao tiếp dữ liệu qua mạng LAN
* Sử dụng Accessibility Service
* Lưu trữ dữ liệu trong SharedPreferences
* Xử lý JSON

### 2.1.2. Phân tích thiết kế cơ chế giao tiếp mạng

Để phân tích thiết kế cơ chế giao tiếp mạng, trước tiên ta cần hiểu rõ về hai giao thức truyền tải thông tin trong mạng máy tính, đó là TCP và UDP.

#### 2.1.2.1. Giao thức TCP

* Giao thức TCP (Transmission Control Protocol - "Giao thức điều khiển truyền vận") là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.
* TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet và các ứng dụng kết quả, trong đó có www, thư điện tử và Secure Shell (SSH).

#### 2.1.2.2. Giao thức UDP

* Giao thức UDP (User Datagram Protocol) là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.
* Những ứng dụng phổ biến sử dụng UDP như DNS (Domain Name System), ứng dụng streaming media, Voice over IP, Trivial File Transfer Protocol (TFTP), và game trực tuyến.

#### 2.1.2.3. Những sự khác nhau giữa hai giao thức TCP và UDP

* Do hai giao thức TCP và UDP được sinh ra để phục vụ những mục đích khác nhau. Cho nên có những khác biệt trong cách truyền tải dữ liệu giữa hai giao thức.
* Một gói tin khi truyền giữa hai máy tính sẽ được phân mảnh ra thành nhiều các mảnh tin (fragments). Mỗi mảnh tin sẽ có trường fragment offset cho biết thứ tự của nó trong gói tin. Dựa vào fragment offset của từng mảnh tin, máy nhận sẽ ghép các mảnh tin lại thành gói tin hoàn chỉnh. Sự khác biệt của hai giao thức TCP và UDP nằm ở cách chúng truyền tải những mảnh tin.
* Giao thức TCP được sử dụng cho những mục đích cần dữ liệu được truyền tải một cách chính xác mà không quá khắt khe về vấn đề độ trễ. TCP đảm bảo rằng máy nhận nhận được toàn bộ các mảnh tin, cũng như các mảnh tin được sắp xếp đúng thứ tự. Vậy nên nó được dùng cho các dịch vụ cần đảm bảo dữ liệu được truyền tải chính xác mà không quá khắt khe về độ trễ như www, email hoặc SSH.
* Giao thức UDP được sử dụng cho những mục đích yêu cầu truyền tải dữ liệu nhanh và ít độ trễ như video call, game trực tuyến. Bù lại, UDP không đảm bảo máy nhận nhận được toàn bộ các mảnh tin cũng như mảnh tin không đảm bảo sắp xếp đúng thứ tự, chỉ liên tục gửi dữ liệu. Đó là lý do thi thoảng khi video call ta thấy hình ảnh hoặc giọng nói đối phương bị giật lag.

#### 2.1.2.4. Phân tích cơ chế giao tiếp mạng cho ứng dụng ClipSync

* Do phần mềm ClipSync hoạt động hoàn toàn trên các máy của người dùng, không có máy chủ nào cho nên phần mềm sẽ sử dụng mô hình giao tiếp ngang hàng (P2P).
* Để các máy giao tiếp với nhau, phần mềm cần một cơ chế để báo hiệu cho nhau biết sự hiện diện của chúng ở trên mạng LAN. Để các phần mềm báo hiệu cho nhau về sự hiện diện của chúng trên mạng LAN, ta sẽ phát tán các gói tin UDP Broadcast, UDP được sử dụng vì giao thức này là một giao thức connectionless, tức là không yêu cầu hai máy phải thiết lập kết nối để có thể truyền tải dữ liệu. Trong đề tài này, các gói tin UDP Broadcast chứa một đoạn text được phát đến tất cả các máy trong mạng và được nghe trên port 7070 khi một máy có dữ liệu muốn các máy khác đồng bộ.
* Đoạn text mà các gói tin UDP Broadcast chứa bao gồm nội dung:

“CS\_BC + Username”. Trong đó:

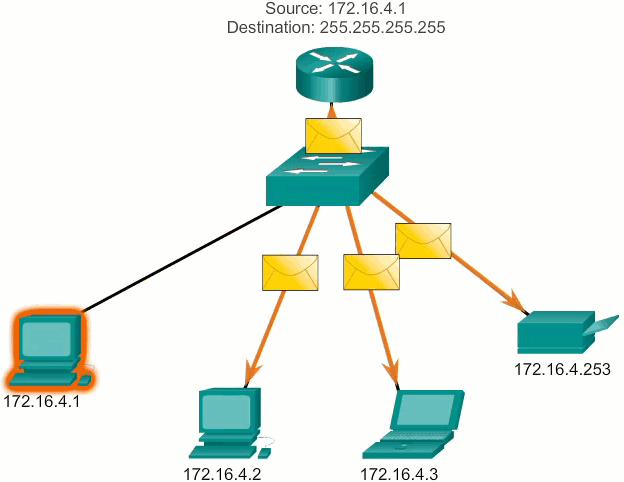
* + “CS\_BC”: Là cờ ClipSync\_Broadcast dùng để các client trên mạng nhận biết đây là một gói tin UDP Broadcast của ứng dụng ClipSync.
  + Username: Là tên người dùng thiết lập trong ứng dụng ClipSync, username được phát kèm theo trong UDP Broadcast để các client khác nhận biết rằng chúng có thuộc cùng về một username của máy đang phát gói tin hay không. Nếu trùng khớp một username, thì các client đó sẽ gửi yêu cầu đồng bộ, nếu không thì chúng sẽ bỏ qua gói tin đó. Điều này nhằm tránh các thiết bị của các người dùng khác nhau nhưng vẫn gửi yêu cầu đồng bộ không cần thiết làm chiếm dụng băng thông mạng.
* Cơ chế hoạt động của UDP Broadcast (ví dụ đối với mạng hình sao):
  + Máy gửi gửi gói UDP Broadcast với địa chỉ đích là địa chỉ Broadcast của mạng

A computer network diagram with a diagram of a computer network

AI-generated content may be incorrect.

Hình 1. UDP Broadcast

* Gói tin được phân phát đến tất cả các máy trong mạng



Hình 2. UDP Broadcast

* Sau khi các máy gửi gói tin UDP Broadcast để thông báo cho các máy khác rằng chúng có dữ liệu cần đồng bộ. Cần có một cơ chế xác thực và gửi dữ liệu giữa các máy. Khi xác thực và gửi dữ liệu giữa các máy cần dữ liệu phải được toàn vẹn cho nên giao thức TCP được sử dụng. Các máy sẽ mở cổng lắng nghe kết nối TCP trên port 7071. Các port 7070 cho UDP Broadcast và 7071 cho TCP là tự chọn.

### 2.1.3. Thiết kế các gói tin

#### 2.1.3.1. Gói tin broadcast

Nội dung gói tin broadcast ở dạng text

|  |  |
| --- | --- |
| Thành phần | Mô tả |
| Cờ “CS\_BC\_” | Viết tắt của ClipSync\_Broadcast, báo hiệu cho phần mềm biết đây là gói tin broadcast. Quá trình phát triển về sau có thể sẽ có thêm các cờ khác |
| Username | Tên người dùng, dùng để phần mềm nhận biết gói tin broadcast thuộc về người dùng nào |
| Nội dung gói tin hoàn chỉnh | |
| (Cờ broadcast) + (Username) | |

#### 2.1.3.2. Gói tin TCP

Nội dung gói tin TCP ở dạng JSON

|  |  |  |
| --- | --- | --- |
| Thuộc tính JSON | Giá trị chấp nhận | Mô tả |
| type | 1. “otp\_auth”  2. “data” | Thuộc tính biểu thị loại dữ liệu, tuỳ thuộc vào thuộc tính type phần mềm sẽ có cách xử lý khác nhau với giá trị của thuộc tính value |
| value | Giá trị chuỗi (string value) | Là chuỗi OTP khi type = “otp\_auth” và là dữ liệu dạng chuỗi khi type = “data” |
| Nội dung gói tin hoàn chỉnh | | |
| {“type”: “type\_here”, “value”: “value\_here”} | | |

### 2.1.4. Phân tích thiết kế cơ chế xác thực OTP

* One-Time Password (OTP) là một phương thức xác thực tạm thời, trong đó mỗi mã OTP chỉ có thể sử dụng một lần và có thời gian ngắn. Để đồng bộ mã OTP giữa các máy ta cần các điều kiện:
* Các máy có password giống nhau
* Các máy có thời gian hệ thống giống nhau
* Để sinh ra mã OTP 6 số từ password và thời gian hệ thống ta sử dụng thuật toán Time-based one-time password (TOTP), mã OTP này sẽ được sử dụng để xác thực giữa các máy với nhau.
* Quy trình xác thực giữa các máy như sau:

A diagram of a host and host

AI-generated content may be incorrect.

Hình 3. Quy trình xác thực OTP

Trong quy trình trên bao gồm hai host A và host B, đầu tiên người dùng thao tác copy nội dung clipboard, host A sẽ phát gói tin UDP Broadcast, trong gói UDP Broadcast có chứa username mà host A thuộc về, host B nhận được gói tin, so sánh username và gửi lại gói tin request chứa OTP. Host A kiểm tra OTP, nếu hợp lệ host A sẽ gửi dữ liệu đến host B.

### 2.1.5. Phân tích thiết kế cơ chế mã hoá dữ liệu

Việc gửi dữ liệu qua mạng mà không mã hoá có thể khiến dữ liệu có thể bị hacker dễ dàng đọc được. Vì vậy cho nên phần mềm cần phải mã hoá dữ liệu trước khi gửi. Để phân tích thiết kế cơ chế mã hoá dữ liệu, trước tiên ta cần hiểu rõ về hai loại thuật toán mã hoá, đó là mã hoá đối xứng và mã hoá bất đối xứng.

#### 2.1.5.1. Thuật toán mã hoá đối xứng

* Thuật toán mã hoá đối xứng là thuật toán mã hoá trong đó khoá dùng cho việc mã hoá và giải mã đều dùng chung một chìa. Do tính chất đó của thuật toán mã hoá đối xứng, các khoá của nó cần phải được giữ bí mật tuyệt đối.
* Một số thuật toán mã hoá đối xứng phổ biến có thể kể đến như AES, DES, Blowfish, vv…

#### 2.1.5.2. Thuật toán mã hoá bất đối xứng

* Thuật toán mã hoá bất đối xứng là thuật toán mã hoá trong đó khoá dùng cho việc mã hoá và giải mã bằng hai chìa khác nhau, một loại chìa là khoá công khai, và loại còn lại là khoá bí mật. Khoá công khai có thể được trao đổi qua môi trường một cách công khai. Tuy nhiên, nhược điểm của loại thuật toán này là có khả năng bị tấn công MITM (Man in the Middle).
* Một số thuật toán mã hoá bất đối xứng phổ biến có thể kể đến như RSA, Diffie-Hellman, vv…

#### 2.1.5.3. Cơ chế mã hoá dữ liệu cho phần mềm ClipSync

* Để các máy có thể mã hoá và giải mã dữ liệu của nhau, ta cần phải giải quyết vấn đề trao đổi khoá giữa các máy. Xét đến thuật toán mã hoá đối xứng, ta nhận thấy rằng do tính chất của thuật toán mã hoá đối xứng, cho nên ta không thể trao đổi khoá qua môi trường mạng vì cơ chế mã và giải mã cùng một chìa, cho nên khoá của thuật toán tuyệt đối giữ bí mật, xét đến thuật toán mã hoá bất đối xứng, tuy rằng khoá công khai của nó có thể được gửi qua môi trường mạng. Nhưng nhược điểm của nó là có thể bị tấn công MITM, dưới đây là biểu đồ mô tả quy trình giao tiếp sử dụng thuật toán RSA khi có và không bị tấn công MITM.
* Ta xét quy trình giao tiếp bình thường:

A diagram of a network

AI-generated content may be incorrect.

Hình 4. Quy trình giao tiếp sử dụng RSA bình thường

* Giải thích chi tiết quy trình trên lần lượt từ trên xuống dưới (quy trình trên phân tích giao tiếp giữa hai host duy nhất trong mạng, thực tế sẽ trong mạng sẽ có nhiều host):

|  |  |  |
| --- | --- | --- |
| Các bước | Gửi từ | Giải thích |
| Broadcast (with Username) | Host A → Host B | Host A gửi UDP Broadcast chứa username đến Host B |
| RSA Public key | Host B → Host A | Host B gửi khoá RSA công khai của nó cho Host A |
| RSA Public key | Host A → Host B | Host A gửi khoá RSA công khai của nó cho Host B |
| Sync request (with OTP) | Host B → Host A | Host B gửi mã OTP đã được mã bằng khoá công khai của Host A cho Host A |
| Clipboard data | Host A → Host B | Dữ liệu clipboard gửi từ Host A tới Host B nếu Host A chấp thuận mã OTP mà Host B gửi tới |

* Nhận xét quy trình trên ta thấy rằng quy trình này là một giao thức đơn giản gồm 5 bước, trong đó hai máy chủ thực hiện các bước trao đổi khóa công khai RSA và sử dụng chúng để mã hóa một mã OTP, từ đó xác thực yêu cầu gửi dữ liệu clipboard. Việc trao đổi khóa RSA công khai cho phép thiết lập một kênh bảo mật mà không cần chia sẻ khóa bí mật trước đó. OTP được mã hóa bằng khóa công khai giúp bảo đảm rằng chỉ Host A (với khóa riêng tư) mới có thể giải mã và xác minh.
* Ta xét quy trình giao tiếp khi bị tấn công MITM:

A diagram of a computer system

AI-generated content may be incorrect.

Hình 5. Quy trình giao tiếp sử dụng RSA bị tấn công MITM

* Trong quy trình trên, dựa vào kỹ thuật tấn công MITM, Attacker có thể xen mình vào giữa giao tiếp của hai Host A và Host B mà hai host này không hề hay biết, Host A sẽ bị Attacker đánh lừa rằng Attacker là Host B và ngược lại. Từ đó Attacker có thể chặn, đọc hoặc thay đổi nội dung gói tin giữa hai Host A và Host B.
* Giải thích chi tiết quy trình trên lần lượt từ trên xuống dưới:

|  |  |  |
| --- | --- | --- |
| Các bước | Gửi từ | Giải thích |
| Broadcast (with Username) | Host A → Attacker  Host A → Host B | Host A gửi UDP Broadcast chứa username đến các máy trong mạng |
| Attacker RSA Public key | Attacker → Host A | Attacker gửi khoá RSA công khai của nó cho Host A |
| Host A RSA Public key | Host A → Attacker | Host A gửi khoá RSA công khai của nó cho Attacker |
| Host B RSA Public key | Host B → Attacker | Host B gửi khoá RSA công khai của nó đến Attacker, Host B nghĩ rằng Attacker là Host A. Attacker chặn gói tin này không gửi đến Host A |
| Attacker RSA Public key | Attacker → Host B | Attacker gửi khoá RSA công khai của nó cho Host B |
| Host B Sync request (with OTP) | Host B → Attacker | Host B gửi mã OTP đã được mã bằng khoá công khai của Attacker cho Attacker. Attacker bắt gói tin này, giải mã và sử dụng OTP này để đánh lừa Host A |
| Attacker Sync request (with OTP from Host B) | Attacker → Host A | Attacker gửi mã OTP đã đánh cắp được từ Host B và mã bằng khoá công khai của Host A cho Host A |
| Clipboard data | Host A → Attacker | Host A xác thực OTP hợp lệ, gửi dữ liệu clipboard được mã bằng khoá công khai của Attacker đến Attacker. Attacker giải mã và khai thác dữ liệu đánh cắp được bằng khoá bí mật của mình |
| Clipboard data | Attacker → Host B | Attacker đóng giả làm Host A, gửi dữ liệu clipboard đến Host B, sử dụng khoá công khai của Host B để mã hoá dữ liệu |

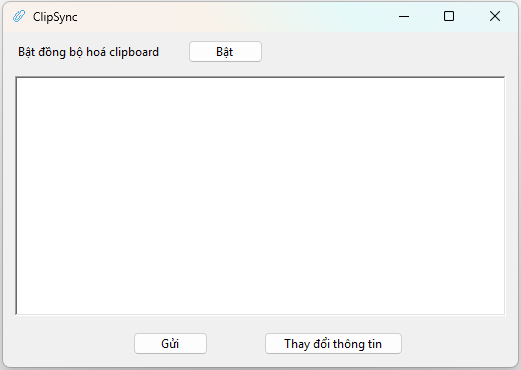
* Nhận xét quy trình trên ta thấy rằng hai Host A và Host B không có cơ chế xác thực khoá công khai của nhau, điều đó dẫn đến Attacker có thể xen ngang và giả mạo khoá một cách dễ dàng. Ngày nay, giao thức bảo mật TLS/SSL thường được sử dụng, trong đó RSA là một thuật toán được sử dụng bên trong các giao thức này. Nó có khả năng xác thực khoá công khai thông qua chứng chỉ, do đó vô hiệu hoá các cuộc tấn công MITM, chứng chỉ của TLS/SSL là tập tin kỹ thuật số được phát hành bởi một tổ chức chứng thực, dùng để:
  + Xác minh danh tính của máy chủ
  + Cung cấp khoá công khai để client mã hoá dữ liệu gửi đến server
  + Kích hoạt kết nối HTTPS an toàn
  + Tuy TLS/SSL rất an toàn cho việc truyền tải dữ liệu, nhưng xét đến ứng dụng ClipSync, ta không có một cơ chế quản lý chứng chỉ phù hợp cho ứng dụng do các host hoạt động theo mô hình P2P, nếu lưu trữ chứng chỉ trong ứng dụng, đó sẽ là lỗ hổng bảo mật lớn vì hacker có thể dễ dàng dịch ngược ứng dụng và lấy được chứng chỉ. Vì vậy, cần một giải pháp khác để đồng bộ khoá trên các host.
  + Vậy sử dụng thuật toán mã hoá bất đối xứng cũng không khả thi, mà mã hoá đối xứng cũng không được vì khoá không thể trao đổi qua mạng. Vậy giải pháp nào cho vấn đề này? Giải pháp khả thi nhất là dẫn xuất khoá từ một thông tin đã biết sẵn giữa các bên, trong phần mềm ClipSync, mật khẩu là thứ giống nhau ở tất cả các bên, tuy nhiên, chỉ sử dụng mật khẩu thô là không đủ an toàn vì chúng thường ngắn, dễ đoán và dễ bị tấn công từ điển hay brute-force. Để tăng cường tính bảo mật, người ta sử dụng một hàm dẫn xuất khoá (Key Derivation Function – KDF), điển hình là PBKDF2 (Password-Based Key Derivation Function 2).
  + PBKDF2 giúp chuyển đổi mật khẩu thô thành một khoá mã hoá mạnh, bằng cách kết hợp với một giá trị salt ngẫu nhiên và lặp lại quá trình băm nhiều lần để làm chậm tốc độ tấn công. Qua đó, hệ thống có thể tạo ra khoá mã hoá một cách an toàn mà không cần phải trao đổi trực tiếp khoá qua mạng, đồng thời tăng cường khả năng chống lại các cuộc tấn công brute-force.
  + Tuy nhiên, do ta cần đồng bộ khoá giữa các máy với nhau, dùng salt ngẫu nhiên sẽ khiến các khoá dẫn xuất khác nhau giữa các máy, vì vậy ta dùng username làm salt cố định cho PBKDF2, do mật khẩu được băm bằng SHA512, lưu trữ nội bộ và dẫn xuất khoá từ giá trị băm khi mở ứng dụng cho nên sử dụng salt cố định không làm giảm tính bảo mật của ứng dụng.

# **CHƯƠNG 3. CHƯƠNG TRÌNH PHẦN MỀM**

## ***3.1. Giao diện***

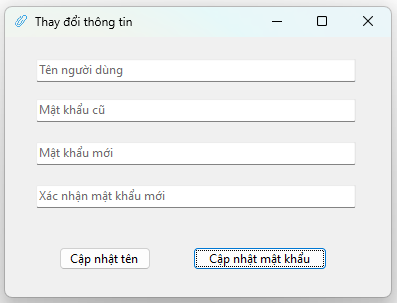
### 3.1.1. Giao diện phía Windows

* Giao diện chính



Hình 6. Giao diện chính phía Windows

* Giao diện thay đổi thông tin



Hình 7. Giao diện thay đổi thông tin phía Windows

### 3.1.2. Giao diện phía Android

* Giao diện chính



Hình 8. Giao diện chính phía Android

* Giao diện thay đổi thông tin



Hình 9. Giao diện thay đổi thông tin phía Android

## ***3.2. Một số đoạn code chính***

### 3.2.1. Android

* **Lớp xử lý UDP Broadcast (BroadcastHandler.java)**

package com.mtechdwork.clipsync;

import android.content.Context;

import android.net.wifi.WifiInfo;

import android.net.wifi.WifiManager;

import android.util.Log;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Objects;

public class BroadcastHandler extends Thread {

private static final int PORT = 7070;

private boolean running = true;

private final Context context;

private final SettingManager settingManager;

private DatagramSocket socket;

BroadcastHandler(Context context) {

this.context = context;

settingManager = new SettingManager(context);

}

private void log(String message, int type) { // Debug method

// Type: 0 - Info, 1 - Warning, 2 - Error

boolean debug = false;

if (!debug) return;

String className = "[Broadcast Handler]";

switch (type) {

case 0:

Log.i(className, message);

break;

case 1:

Log.w(className, message);

break;

case 2:

Log.e(className, message);

}

}

public void stopListening() {

running = false;

if (socket != null && !socket.isClosed()) socket.close();

}

private String getIPAddress() {

WifiManager wifiManager = (WifiManager) context.getApplicationContext().getSystemService(Context.WIFI\_SERVICE);

if (wifiManager != null) {

WifiInfo wifiInfo = wifiManager.getConnectionInfo();

int ipAddress = wifiInfo.getIpAddress();

// Chuyển đổi từ int sang địa chỉ IP

return (ipAddress & 0xFF) + "." +

((ipAddress >> 8) & 0xFF) + "." +

((ipAddress >> 16) & 0xFF) + "." +

((ipAddress >> 24) & 0xFF);

}

return null;

}

private boolean checkSenderMatch(String receivedMessage) {

String broadcastContent = "CS\_BC\_" + settingManager.getUsername();

return receivedMessage.equals(broadcastContent);

}

@Override

public void run() {

try {

socket = new DatagramSocket(PORT);

byte[] buffer = new byte[1024];

while (running) {

DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

socket.receive(packet);

String receivedMessage = new String(packet.getData(), 0, packet.getLength());

InetAddress senderAddress = packet.getAddress();

if (checkSenderMatch(receivedMessage) && !Objects.equals(senderAddress.getHostAddress(), getIPAddress())) {

log("Received: " + receivedMessage + " from " + senderAddress.getHostAddress(), 0);

Communication communication = new Communication(context);

communication.sendSyncRequest(senderAddress);

}

}

if (socket != null) socket.close();

} catch (Exception e) {

log(e.getMessage(), 2);

stopListening();

}

log("Thread stopped!", 0);

}

@Override

public void interrupt() {

super.interrupt();

log("Thread interrupt!", 0);

stopListening();

}

}

* **Lớp mã hoá giải mã XchaCha20 – Poly1305 (XchaChaCrypto.java)**

package com.mtechdwork.clipsync;  
  
import android.content.Context;  
import android.util.Base64;  
import android.util.Log;  
  
import com.google.crypto.tink.Aead;  
import com.google.crypto.tink.KeysetHandle;  
import com.google.crypto.tink.aead.AeadConfig;  
import com.google.crypto.tink.aead.XChaCha20Poly1305KeyManager;  
import com.google.crypto.tink.subtle.XChaCha20Poly1305;  
  
import java.nio.charset.StandardCharsets;  
import java.security.GeneralSecurityException;  
  
public class XChaChaCrypto {  
 private static Aead *aead*;  
  
 public static void init(Context context) {  
 if (!*isTinkInitialized*()) {  
 try {  
 AeadConfig.*register*(); // Đăng ký cấu hình Tink  
  
 *loadKey*(context);  
 } catch (Exception e) {  
 *log*(e.getMessage(), 2);  
 }  
 }  
 }  
  
 */\*\* @noinspection SameParameterValue\*/* private static void log(String message, int type) {  
 // Type: 0 - Info, 1 - Warning, 2 - Error  
 boolean debug = false;  
 if (!debug) return;  
 String className = "[XChaChaCrypto]";  
 switch (type) {  
 case 0:  
 Log.*i*(className, message);  
 break;  
  
 case 1:  
 Log.*w*(className, message);  
 break;  
  
 case 2:  
 Log.*e*(className, message);  
 }  
 }  
  
 public static boolean isTinkInitialized() {  
 try {  
 //noinspection unused  
 KeysetHandle testKeyset = KeysetHandle.*generateNew*(XChaCha20Poly1305KeyManager.*xChaCha20Poly1305Template*());  
 return true;  
 } catch (GeneralSecurityException e) {  
 return false;  
 }  
 }  
  
 public static void loadKey(Context context) {  
 new Thread(() -> {  
 byte[] derivedKey;  
 try {  
 derivedKey = PBKDF2.*deriveKey*(context);  
 *aead* = new XChaCha20Poly1305(derivedKey);  
 } catch (Exception e) {  
 *log*(e.getMessage(), 2);  
 }  
 }).start();  
 }  
  
 public static String encrypt(String plaintext, byte[] associatedData) {  
 try {  
 return Base64.*encodeToString*(*aead*.encrypt(plaintext.getBytes(StandardCharsets.*UTF\_8*), associatedData), Base64.*NO\_WRAP*);  
 } catch (GeneralSecurityException e) {  
 *log*(e.getMessage(), 2);  
 }  
 return null;  
 }  
  
 public static String decrypt(String ciphertext, byte[] associatedData) {  
 byte[] decryptedBytes = null;  
 try {  
 decryptedBytes = *aead*.decrypt(Base64.*decode*(ciphertext, Base64.*NO\_WRAP*), associatedData);  
 } catch (GeneralSecurityException e) {  
 *log*(e.getMessage(), 2);  
 }  
 return new String(decryptedBytes, StandardCharsets.*UTF\_8*);  
 }  
}

* **Lớp xử lý xác thực và tạo mã OTP (Authenticator.java)**

package com.mtechdwork.clipsync;  
  
import android.annotation.SuppressLint;  
import android.content.Context;  
import android.util.Log;  
  
import javax.crypto.Mac;  
import javax.crypto.spec.SecretKeySpec;  
  
public class Authenticator {  
  
 private final SettingManager settingManager;  
  
 Authenticator(Context context) {  
 settingManager = new SettingManager(context);  
 }  
  
 */\*\* @noinspection SameParameterValue\*/* private void log(String message, int type) {  
 // Type: 0 - Info, 1 - Warning, 2 - Error  
 boolean debug = false;  
 if (!debug) return;  
 String className = "[Accessibility Service]";  
 switch (type) {  
 case 0:  
 Log.*i*(className, message);  
 break;  
  
 case 1:  
 Log.*w*(className, message);  
 break;  
  
 case 2:  
 Log.*e*(className, message);  
 }  
 }  
  
 @SuppressLint("DefaultLocale")  
 public String genOTP() {  
 long currentTime = System.*currentTimeMillis*() / 30000; // Mỗi 30 giây đổi OTP  
 String timeHex = Long.*toHexString*(currentTime).toUpperCase();  
  
 try {  
 // HMAC-SHA1  
 String passHash = settingManager.getPassword();  
  
 if (passHash.isEmpty()) return "";  
  
 byte[] keyBytes = passHash.getBytes();  
 SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "HmacSHA1");  
 Mac mac = Mac.*getInstance*("HmacSHA1");  
 mac.init(keySpec);  
  
 byte[] hash = mac.doFinal(timeHex.getBytes());  
  
 // Dùng thuật toán để trích xuất 6 chữ số OTP  
 int offset = hash[hash.length - 1] & 0xF;  
 int otp = ((hash[offset] & 0x7F) << 24) |  
 ((hash[offset + 1] & 0xFF) << 16) |  
 ((hash[offset + 2] & 0xFF) << 8) |  
 (hash[offset + 3] & 0xFF);  
 otp %= 1000000; // Chỉ lấy 6 số cuối  
  
 return String.*format*("%06d", otp);  
 } catch (Exception e) {  
 log(e.getMessage(), 2);  
 }  
 return "";  
 }  
  
 public boolean checkOTPMatch(String otp) {  
 return otp.equals(genOTP());  
 }  
}

### 3.2.2. Windows

* **Lớp xử lý UDP Broadcast (BroadcastHandler.vb)**

Imports System.Net

Imports System.Net.Sockets

Imports System.Text

Imports System.Threading

Public Class BroadcastHandler

Private broadcastListening As Thread

Private listener As UdpClient

Private running = False

Private UDP\_PORT = 7070

Public Sub startListening()

running = True

listener = New UdpClient(New IPEndPoint(LocalIP.getIPAddress(), UDP\_PORT))

broadcastListening = New Thread(AddressOf listening)

broadcastListening.IsBackground = True

broadcastListening.Start()

End Sub

Public Sub stopListening()

running = False

If listener IsNot Nothing Then listener.Close()

End Sub

Private Function checkSenderMatch(receivedMessage As String) As Boolean

Dim broadcastContent = "CS\_BC\_" + SettingManager.getUsername()

Return receivedMessage = broadcastContent

End Function

Private Sub listening()

ClipSyncDebug.log("[BroadcastHandler] Start new thread")

Dim remoteEP As New IPEndPoint(IPAddress.Any, 0)

While running

Try

Dim data As Byte() = listener.Receive(remoteEP)

Dim message As String = Encoding.UTF8.GetString(data)

If checkSenderMatch(message) AndAlso LocalIP.getIPAddress().ToString <> remoteEP.Address.ToString Then

ClipSyncDebug.log("Received from " & remoteEP.Address.ToString() & ": " & message)

Communication.sendSyncRequest(remoteEP.Address)

End If

Catch ex As Exception

ClipSyncDebug.log("[BroadcastHandler] UDP receive error: " & ex.Message)

End Try

End While

ClipSyncDebug.log("[BroadcastHandler] Stop thread")

End Sub

End Class

* **Lớp mã hoá giải mã XchaCha20 – Poly1305 (XchaChaCrypto.vb)**

Imports System.Security.Cryptography

Imports System.Text

Imports Sodium

Public Class XChaChaCrypto

Private Shared pbkdf2Key As Byte()

Public Shared Sub loadKey()

pbkdf2Key = PBKDF2.deriveKey()

End Sub

Private Shared Function CreateNonce(length As Integer) As Byte()

Dim nonce(length - 1) As Byte

RandomNumberGenerator.Fill(nonce)

Return nonce

End Function

Public Shared Function encrypt(plaintext As String, associatedData As Byte()) As String

Dim plainBytes As Byte() = Encoding.UTF8.GetBytes(plaintext)

Dim nonce = CreateNonce(24) ' XChaCha20 yêu cầu nonce dài 24 bytes

Dim cipherBytes = SecretAeadXChaCha20Poly1305.Encrypt(plainBytes, nonce, pbkdf2Key, associatedData)

Dim finalOutput = nonce.Concat(cipherBytes).ToArray()

Return Convert.ToBase64String(finalOutput)

End Function

Public Shared Function decrypt(ciphertext As String, associatedData As Byte()) As String

Dim cipherBytes = Convert.FromBase64String(ciphertext)

' Trích xuất nonce và ciphertext

Dim extractedNonce = cipherBytes.Take(24).ToArray()

Dim actualCipher = cipherBytes.Skip(24).ToArray()

Dim decryptedBytes = SecretAeadXChaCha20Poly1305.Decrypt(actualCipher, extractedNonce, pbkdf2Key, associatedData)

Return Encoding.UTF8.GetString(decryptedBytes)

End Function

End Class

* **Lớp xử lý xác thực và tạo mã OTP (Authenticator.vb)**

Imports System.Security.Cryptography

Imports System.Text

Public Class Authenticator

Public Shared Function genOTP() As String

Dim currentTime As Long = DateTimeOffset.UtcNow.ToUnixTimeMilliseconds() \ 30000 ' Every 30 seconds

Dim timeHex As String = currentTime.ToString("X").ToUpper()

Try

' Retrieve the password

Dim passHash As String = SettingManager.getPassword()

If String.IsNullOrEmpty(passHash) Then Return ""

Dim keyBytes As Byte() = Encoding.UTF8.GetBytes(passHash)

Dim mac As New HMACSHA1(keyBytes)

Dim hash As Byte() = mac.ComputeHash(Encoding.UTF8.GetBytes(timeHex))

' Extract 6-digit OTP

Dim offset As Integer = hash(hash.Length - 1) And &HF

Dim otp As Integer = ((hash(offset) And &H7F) << 24) Or

((hash(offset + 1) And &HFF) << 16) Or

((hash(offset + 2) And &HFF) << 8) Or

(hash(offset + 3) And &HFF)

otp = otp Mod 1000000 ' Keep only the last 6 digits

Return otp.ToString("D6")

Catch ex As Exception

ClipSyncDebug.log(ex.Message)

End Try

Return ""

End Function

Public Shared Function checkOTPMatch(otp As String) As Boolean

Return genOTP() = otp

End Function

End Class

## ***3.3. Những hạn chế của chương trình***

# **CHƯƠNG 4. KẾT LUẬN**

# **TÀI LIỆU THAM KHẢO**

* <https://developer.android.com/about/versions/10/privacy/changes?hl=vi#clipboard-data>
* <https://viblo.asia/p/lap-trinh-android-voi-bo-nho-trong-internal-storage-PdbknoJRvyA>
* <https://stackoverflow.com/questions/64030969/how-to-broadcast-a-udp-packet-from-android>
* <https://developers.google.com/tink?hl=vi>
* <https://vi.wikipedia.org/wiki/TCP#:~:text=TCP%20(Transmission%20Control%20Protocol%20%2D%20%22,li%E1%BB%87u%20ho%E1%BA%B7c%20c%C3%A1c%20g%C3%B3i%20tin>.
* <https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.sha512?view=net-9.0>
* <https://learn.microsoft.com/en-us/dotnet/api/system.net.sockets.socket?view=net-9.0>
* <https://learn.microsoft.com/en-us/dotnet/api/system.net.sockets.tcplistener?view=net-9.0>