BAN CƠ YẾU CHÍNH PHỦ

**HỌC VIỆN KỸ THUẬT MẬT MÃ**


# ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**ĐỀ TÀI**

## XÂY DỰNG HỆ THỐNG SAO LƯU, PHỤC HỒI DỮ LIỆU CÓ MÃ HÓA ĐẦU CUỐI

Sinh viên thực hiện: Trần Quang Chung

Khóa: CT4

Chuyên ngành: Kỹ thuật phần mềm nhúng và di động

Mã ngành: 748.02.01

Giảng viên hướng dẫn: TS. Phạm Văn Hưởng

**HÀ NỘI - 2024**

BAN CƠ YẾU CHÍNH PHỦ

**HỌC VIỆN KỸ THUẬT MẬT MÃ**


# ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**ĐỀ TÀI**

**XÂY DỰNG HỆ THỐNG SAO LƯU, PHỤC HỒI DỮ LIỆU CÓ MÃ HÓA ĐẦU CUỐI**

Sinh viên thực hiện: Trần Quang Chung

Khóa: CT4

Chuyên ngành: Kỹ thuật phần mềm nhúng và di động

Mã ngành: 748.02.01

Giảng viên hướng dẫn: TS. Phạm Văn Hưởng

**HÀ NỘI - 2024**

# LỜI CẢM ƠN

Sau thời gian học tập và rèn luyện tại Học viện Kỹ thuật Mật mã, giờ đến lúc những kiến thức của em được vận dụng vào thực tiễn công việc. Em lựa chọn đồ án tốt nghiệp đề tài: " *Xây dựng hệ thông sao lưu, phục hồi dữ liệu có mã hóa đầu cuối* " để giúp em củng cố và học tập, rèn luyện thêm về những kĩ năng, kiến thức để trang bị cho tương lai. Có được thành công này, ngoài sự nỗ lực học hỏi của bản thân còn có sự hướng dẫn tận tình của các thầy cô, các anh chị và các bạn trong trường.

Lời đầu tiên em xin gửi lời cảm ơn chân thành đến Ban Giám đốc Trường Học viện Kỹ thuật Mật mã và các quý thầy cô trong trường nói chung và Khoa Công nghệ thông tin nói riêng, đã tận tình chỉ dạy, truyền đạt những kiến thức, kinh nghiệm cho em từ đó giúp em có được những kiến thức cũng như kĩ năng cần thiết hình thành nên nền tảng nghề nghiệp cơ bản.

Đặc biệt em xin gửi lời cảm ơn chân thành tới giảng viên hướng dẫn, TS. Phạm Văn Hưởng, người đã tận tình hướng dẫn và cho em những định hướng và kiến thức quý báu để thực hiện và hoàn thành đề tài.

Cùng với đó em xin cảm ơn đến những người thân, bạn bè đã đồng hành và hỗ trợ em hoàn thành đồ án này.

Dù đã rất cố gắng, tuy nhiên kiến thức chuyên môn còn hạn chế và bản thân còn thiếu kinh nghiệm thực tiễn, nên nội dung báo cáo không tránh khỏi những khiếm khuyết, em rất mong có những ý kiến và giúp đỡ của quý thầy cô để báo cáo này có thể hoàn thiện hơn.

Cuối cùng, em xin kính chúc quý thầy cô, anh chị, bạn bè dồi dào sức khỏe và thành công trong sự nghiệp, gặp được nhiều điều tốt đẹp trong cuộc sống!

Em xin trân trọng cảm ơn!

# LỜI CAM ĐOAN

Em xin cam đoan bản đồ án này do em tự nghiên cứu dưới sự hướng dẫn của thầy giáo TS. Phạm Văn Hưởng.

Để hoàn thành đồ án này, em chỉ sử dụng những tài liệu đã ghi trong mục tài liệu tham khảo, ngoài ra không sử dụng bất cứ tài liệu nào khác mà không được ghi.

Nếu sai, em xin chịu mọi hình thức kỷ luật theo quy định của Học viện.

*Hà Nội, ngày 7 tháng 6 năm 2024*

**Sinh viên thực hiện**

(*Ký tên và ghi rõ họ tên*)

**Trần Quang Chung**

# MỤC LỤC

# DANH MỤC BẢNG BIỂU

# DANH MỤC HÌNH VẼ

# LỜI NÓI ĐẦU

Thị trường Việt Nam đang phát triển mạnh mẽ nhưng cũng đầy rủi ro, đặc biệt là trong các lĩnh vực đòi hỏi sự liên tục và an toàn dữ liệu như tài chính, kế toán và chứng khoán. Trong thời đại công nghệ số, máy tính trở thành công cụ không thể thiếu, chứa đựng nhiều dữ liệu quan trọng như file, ảnh và các dữ liệu khác. Tuy nhiên, các sự cố hỏng ổ cứng, lỗi người dùng, hoặc tấn công bởi có virus có thể khiến dữ liệu này bị mất. Điều này dẫn đến nhu cầu cấp thiết về một giải pháp sao lưu và phục hồi dữ liệu hiệu quả.

Đối với các tổ chức doanh nghiệp, theo báo cáo của Symantec, một lượng lớn doanh nghiệp đã từng trải qua tình trạng mất dữ liệu và buộc phải sử dụng các bản sao lưu để khôi phục nó. Thiên tai, tấn công mạng và lỗi con người là những nguyên nhân phổ biến dẫn đến mất dữ liệu. Đáng chú ý là 96% các máy trạm không được bảo vệ bằng các giải pháp sao lưu, khiến cho việc phục hồi dữ liệu trở nên khó khăn và đắt đỏ. Doanh nghiệp có thể chịu tổn thất nặng nề và thậm chí đóng cửa nếu không có biện pháp phòng ngừa hiệu quả.

Chính vì vậy, đề tài "Xây dựng hệ thống sao lưu, phục hồi dữ liệu có mã hóa đầu cuối" được đưa ra nhằm đề xuất một giải pháp dựa trên công nghệ hiện đại, với ứng dụng của giao thức S3 và phần mềm Duplicati. Đề tài được chia thành ba chương chính:

**Chương 1**: Tổng quan về sao lưu dữ liệu và mã hóa đầu cuối.

**Chương 2**: Phân tích và thiết kế hệ thống.

**Chương 3**: Xây dựng hệ thống và thực nghiệm.

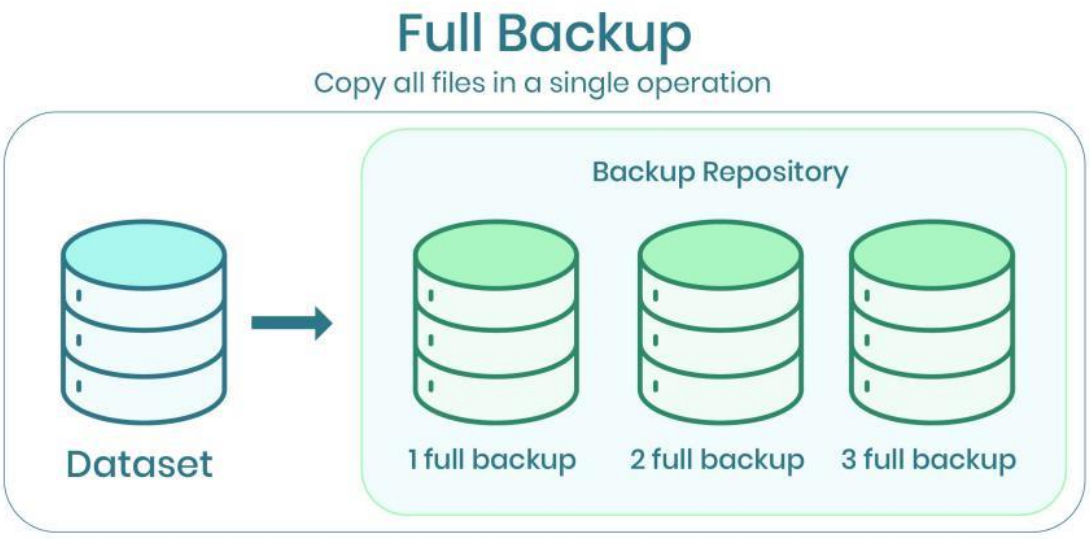
Với sự giới hạn về kinh nghiệm và kiến thức, đề tài này chắc chắn còn nhiều thiếu sót và rất mong nhận được ý kiến đóng góp từ thầy cô trong hội đồng, để em có thể hoàn thành hơn trong đồ án tốt nghiệp của mình.

# CHƯƠNG 1. TỔNG QUAN VỀ SAO LƯU DỮ LIỆU VÀ MÃ HÓA ĐẦU CUỐI

## 1.1. Tổng quan về sao lưu và lưu trữ

Sao lưu (backup) là quá trình tạo ra các bản sao dữ liệu từ nguồn gốc để đảm bảo tính an toàn và khả năng khôi phục dữ liệu trong trường hợp mất mát hoặc xảy ra sự cố. Đây là phần quan trọng của quản lý dữ liệu và thông tin, giúp bảo vệ khỏi rủi ro như lỗi phần cứng, phần mềm, tấn công máy tính hoặc thậm chí là thiên tai.Tuy nhiên, nhiệm vụ backup này không hề dễ dàng do khối lượng dữ liệu và chi phí thiết bị lưu trữ đều ở mức cao, đó là lý do tại sao có nhiều chiến lược sao lưu giúp tối ưu hoá việc lưu trữ dữ liệu mà không tốn nhiều chi phí. Các chiến lược backup phổ biến có thể kể đến sao lưu toàn bộ, sao lưu gia tăng, sao lưu khác biệt.

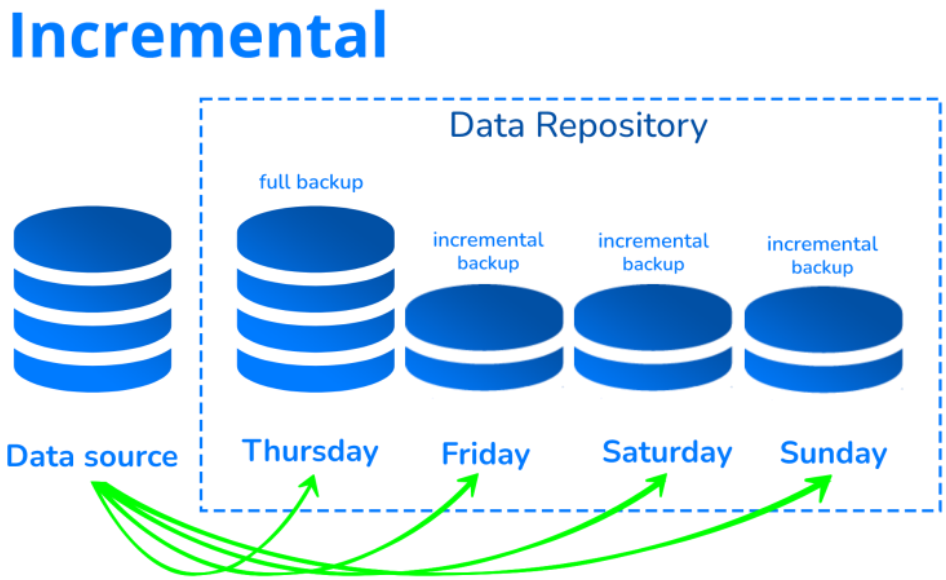
### 1.1.1 Sao lưu toàn bộ (Full Backup)




*Hình 1.1 Sao lưu đầy đủ (Full Backup)*

Sao lưu toàn bộ (Full Backup) là tạo ra bản sao lưu mọi thứ người dùng muốn được bảo vệ. Mặc dù sao lưu đầy đủ được cho là cung cấp việc bảo vệ sự toàn vẹn tốt nhất cho dữ liệu, nhưng hầu hết người dùng không sử dụng phương thức sao lưu dữ liệu này hàng ngày vì nếu dữ liệu lớn thì thời gian backup rất lâu, làm tốn bộ nhớ lưu trữ và chi phí đầu tư.

### 1.1.2 Sao lưu gia tăng (Incremental Backup)

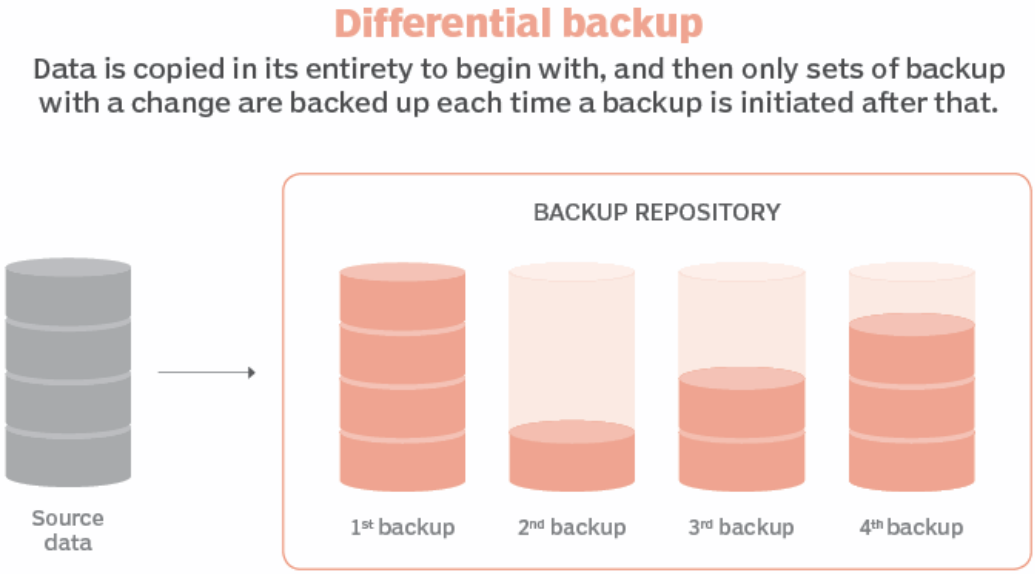



*Hình 1.2 Sao lưu gia tăng ( Incremental Backup)*

Sao lưu gia tăng (Incremental Backup) là dạng phương thức sao lưu cần ít nhất một bản sao lưu đầy đủ. Sau đó, chỉ có những dữ liệu đã thay đổi từ lần sao lưu đầy đủ cuối cùng sẽ được sao lưu lại. Sao lưu gia tăng không mất quá nhiều thời gian và bộ nhớ so với sao lưu toàn bộ. Mặc dù có thể tăng tiến về tốc độ lưu trữ nhưng ở mặt ngược lại thì thời gian khôi phục khá lâu. Giả sử người dùng muốn khôi phục bản sao lưu ở ngày thứ bảy. Để làm điều đó, trước tiên người dùng phải khôi phục bản sao lưu đầy đủ của ngày đầu tiên, sau đó khôi phục bản sao lưu của ngày thứ năm, tiếp theo đó là ngày thứ sáu. Không thể hoàn toàn khôi phục dữ liệu nếu một trong các bản sao lưu gia tăng bị thiếu hoặc hỏng.

Một số biến thể của sao lưu tăng tiến có thể kể đến sao lưu đầy đủ tổng hợp (synthetic full backup) và sao lưu tăng tiến liên tục (incremental-forever backup).

### 1.1.3 Sao lưu khác biệt (Differential Backup)




*Hình 1.3 Sao lưu khác biệt (Differential Backup)*

Sao lưu khác biệt (differential backup) giống với sao lưu tăng tiến ở việc nó bắt đầu bằng một bản sao lưu đầy đủ và các bản sao lưu tiếp theo chỉ chứa dữ liệu đã thay đổi. Sự khác biệt chủ yếu giữa sao lưu khác biệt so với sao lưu tăng tiến là trong khi sao lưu tăng tiến chỉ bao gồm dữ liệu đã thay đổi kể từ lần sao lưu trước, thì sao lưu khác biệt chứa tất cả dữ liệu đã thay đổi kể từ lần sao lưu đầy đủ cuối cùng.

Giả sử rằng người dùng muốn tạo một bản sao lưu đầy đủ vào ngày thứ nhất và sử dụng sao lưu khác biệt cho những ngày còn lại. Bản sao lưu của ngày thứ hai sẽ chứa tất cả dữ liệu đã thay đổi kể từ ngày thứ nhất, lúc này nó sẽ giống với một bản sao lưu tăng tiến. Tuy nhiên, vào ngày thứ ba, bản sao lưu khác biệt cũng sẽ sao lưu lại mọi dữ liệu đã thay đổi kể từ ngày đầu tiên, bao gồm cả những thay đổi trong ngày thứ hai.

Ưu điểm mà sao lưu khác biệt mang lại so với sao lưu tăng tiến là thời gian khôi phục dữ liệu sẽ ngắn hơn. Khôi phục bản sao lưu khác biệt không bao giờ yêu cầu nhiều hơn hai bản sao lưu, một bản sao lưu đầy đủ và một bản sao lưu khác biệt tại thời điểm đó, trong khi sao lưu tăng tiến có thể yêu

cầu một số lượng lớn các bản sao lưu. Tuy nhiên, sao lưu khác biệt yêu cầu một dung lượng lưu trữ lớn hơn.

Một số hệ thống sao lưu có sử dụng sao lưu khác biệt bao gồm:

Veritas Backup Exec

NovaBACKUP

Acronis True Image

Microsoft Azure Backup

Commvault

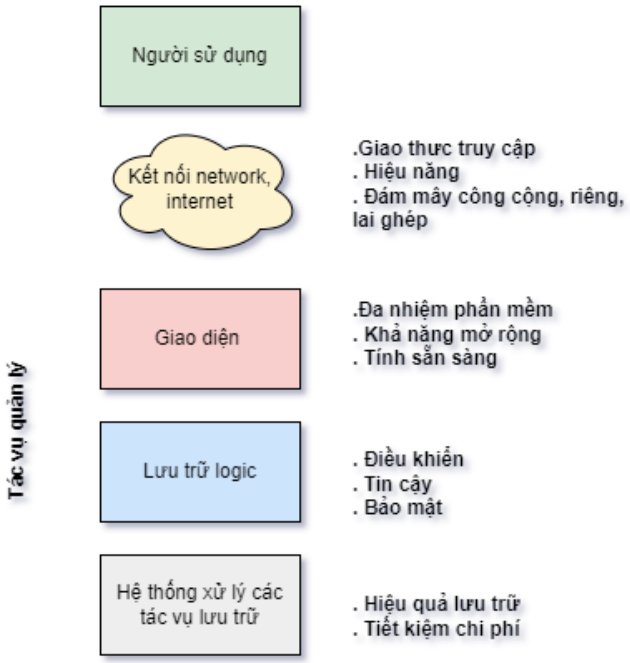
Hầu hết các hệ thống trên đều hỗ trợ cả sao lưu tăng tiến và khác biệt.

### 1.1.4 Kiến trúc lưu trữ đám mây (Cloud Storage)

Khác với lưu trữ truyền thống bao gồm sử dụng ổ đĩa cứng, máy chủ và thiết bị lưu trữ vật lý thì lưu trữ đám mây theo định nghĩa của Google:"*Lưu trữ đám mây là một chế độ lưu trữ dữ liệu máy tính trong đó dữ liệu kỹ thuật số được lưu trữ trên các máy chủ ở các vị trí bên ngoài. Các máy chủ được duy trì bởi nhà cung cấp bên thứ ba chịu trách nhiệm lưu trữ, quản lý và bảo mật dữ liệu được lưu trữ trên cơ sở hạ tầng của họ. Nhà cung cấp đảm bảo rằng dữ liệu trên máy chủ của họ luôn có thể truy cập được thông qua kết nối internet công cộng hoặc riêng tư. Lưu trữ đám mây cho phép các tổ chức lưu trữ, truy cập và duy trì dữ liệu để họ không cần sở hữu và vận hành trung tâm dữ liệu của riêng mình, chuyển chi phí từ mô hình chi tiêu vốn sang hoạt động. Lưu trữ đám mây có thể mở rộng, cho phép các tổ chức mở rộng hoặc giảm dung lượng dữ liệu của họ tùy theo nhu cầu*"

Hiện nay, lưu trữ đám mây được rất nhiều tổ chức và cá nhân sử dụng. Hệ thống lưu trữ đám mây áp dụng các chiến lược sao lưu một cách linh hoạt mang tới sự linh hoạt, khả năng điều chỉnh quy mô dung lượng dữ liệu theo nhu cầu và tính bền lâu. Kiến trúc lưu trữ đám mây cung cấp một giao diện người dùng xuất API để truy cập vào bộ lưu trữ. Đằng sau giao diện người dùng là phần mềm trung gian gọi là logic lưu trữ. Lớp này thực hiện nhiều

tính năng như sao chép và giảm dữ liệu… Cuối cùng, phần cuối thực hiện lưu trữ vật lý cho dữ liệu.




*Hình 1.4 Kiến trúc tổng thể lưu trữ đám mây*

**Khả năng quản lý**: Một trọng tâm chính của lưu trữ đám mây là chi phí. Nếu khách hàng có thể mua và quản lý dung lượng lưu trữ cục bộ so với việc thuê nó trên đám mây, thì thị trường lưu trữ đám mây sẽ biến mất. Nhưng chi phí có thể được chia thành hai loại cấp cao: chi phí của chính hệ sinh thái lưu trữ vật lý và chi phí quản lý nó. Chi phí quản lý là ẩn nhưng đại diện cho một thành phần dài hạn của tổng chi phí. Vì lý do này, lưu trữ đám mây phải tự quản lý ở mức độ lớn. Khả năng giới thiệu bộ lưu trữ mới trong đó hệ thống tự động cấu hình để chứa nó và khả năng tìm và tự phục hồi khi có lỗi là rất quan trọng. Các khái niệm như điện toán tự trị sẽ có vai trò chính trong kiến trúc lưu trữ đám mây trong tương lai

**Phương thức truy cập**: Một trong những điểm khác biệt nổi bật nhất giữa lưu trữ đám mây và lưu trữ truyền thống là phương tiện mà nó được truy cập. Hầu hết các nhà cung cấp triển khai nhiều phương pháp truy cập, nhưng API dịch vụ web là phổ biến. Nhiều API được triển khai dựa trên các nguyên tắc REST, ngụ ý một sơ đồ dựa trên đối tượng được phát triển trên HTTP (sử dụng HTTP làm phương tiện vận chuyển).

**Hiệu năng**: Có nhiều khía cạnh đối với hiệu suất, nhưng khả năng di chuyển dữ liệu giữa người dùng và nhà cung cấp dịch vụ lưu trữ đám mây từ xa là thách thức lớn nhất đối với dịch vụ lưu trữ đám mây. Vấn đề, cũng là đặc điểm của Internet, là TCP. TCP kiểm soát luồng dữ liệu dựa trên các xác nhận gói từ điểm cuối ngang hàng. Mất gói hoặc đến muộn, cho phép kiểm soát tắc nghẽn, hạn chế hơn nữa hiệu suất để tránh các sự cố mạng toàn cầu hơn. TCP lý tưởng cho việc di chuyển một lượng nhỏ dữ liệu qua Internet toàn cầu nhưng ít phù hợp hơn cho việc di chuyển dữ liệu lớn hơn, với thời gian khứ hồi (RTT) ngày càng tăng.

**Kiểm soát dữ liệu**: Khả năng của khách hàng trong việc kiểm soát và quản lý cách dữ liệu của họ được lưu trữ và các chi phí liên quan đến dữ liệu đó là rất quan trọng. Nhiều nhà cung cấp dịch vụ lưu trữ đám mây triển khai các biện pháp kiểm soát giúp người dùng kiểm soát tốt hơn chi phí của họ. Amazon triển khai giảm dự phòng lưu trữ (RRS) để cung cấp cho người dùng phương tiện giảm thiểu chi phí lưu trữ tổng thể. Dữ liệu được sao chép trong cơ sở hạ tầng Amazon S3, nhưng với RRS, dữ liệu được sao chép ít lần hơn với khả năng mất dữ liệu. Điều này lý tưởng cho dữ liệu có thể được tạo lại hoặc có bản sao tồn tại ở nơi khác.

## 1.2. Phát biểu bài toán

Với sự phát triển mạnh mẽ của công nghệ thông tin, thì sao lưu và lưu trữ dữ liệu là vấn đề thiết yếu đối với mỗi cá nhân hay doanh nghiệp. Máy tính hay các máy chủ hiện đại tới đâu thì cũng luôn tiềm ẩn các nguy cơ rủi ro chẳng hạn như: máy nhiễm mã độc, người dùng xóa nhầm file, ổ cứng hỏng đột ngột, các yếu tố thiên tai như hỏa hoạn,…chính vì thế cần phải sao lưu dữ liệu để phòng ngừa các nguy cơ có thể xảy ra. Có một số cách sao lưu thường sử dụng như:

+ Sử dụng thiết bị lưu trữ ngoài: USB, ổ cứng ngoài, ổ cứng di động,..

+ Sử dụng các nền đảng đám mây lưu trữ

Cả hai cách đều thực hiện thủ công và không có gì đảm bảo và chắc chắn rằng người dùng sẽ có động lực để thực hiện chúng đầy đủ.

Ngoài vấn đề sao lưu dữ liệu còn vấn đề khác là tính bảo mật của dữ liệu khi tiến hành sao lưu cũng rất quan trọng, cả hai phương pháp trên đề copy file, chỉ cần người khác tiếp cận được file thì họ sẽ đọc được dữ liệu. Với những phân tích như trên việc phát triển một giải pháp sao lưu dữ liệu liệu khắc phục được những nhược điểm trên là cần thiết.

Vì vậy, đề tài nghiên cứu giao thức S3 và xây dựng hệ thống sao lưu tự động và phục hồi dữ liệu tích hợp mã hóa đầu cuối sẽ giúp cho các tổ chức, doanh nghiệp và cá nhân có thể lưu trữ dữ liệu một cách an toàn, đồng thời đảm bảo khả năng phục hồi dữ liệu nhanh chóng và hiệu quả khi có sự cố xảy ra.

Mục tiêu của đề tài là nghiên cứu và phân tích các yêu cầu và tiêu chuẩn của hệ thống sao lưu và phục hồi dữ liệu, xác định các giải pháp sao lưu và phục hồi dữ liệu trên giao thức S3 và xây dựng một hệ thống tin cậy, đáp ứng được các yêu cầu về tính bảo mật và khả năng phục hồi dữ liệu.

Đối tượng nghiên cứu của đề tài: Giao thức S3, Duplicati, Mã hóa đầu cuối (RSA, AES)

Phạm vi nghiên cứu của đề tài: Nghiên cứu và phân tích kiến trúc mã nguồn, các module xử lý S3 của Duplicati,

Một số tính năng chính cần có:

- Chỉ định được dữ liệu cần sao lưu
- Chỉ định giao thức S3 và nền tảng đám mây lưu trữ Amazon Web Services.
- Sử dụng chính tài khoản của người dùng trên nền tảng đám mây để sao lưu.
- Mã hóa dữ liệu trước khi đẩy lên đám mây lưu trữ
- Giải mã dữ liệu về ban đầu khi thực hiện khôi phục dữ liệu từ đám mây lưu trữ.

### 1.2.1 Mô hình và giải pháp tổng thể

Như đã phân tích ở nội dung trên cần xây dựng một giải pháp sao lưu dữ liệu tại máy trạm sử dụng các nền tảng đám mây công cộng. Để đảm bảo an toàn bảo mật thì trước khi dữ liệu được đẩy lên cloud, chúng phải được mã hóa.

Các giai đoạn trong bài toán này:

- Bước 1: Xác định và quản lý dữ liệu cần sao lưu.

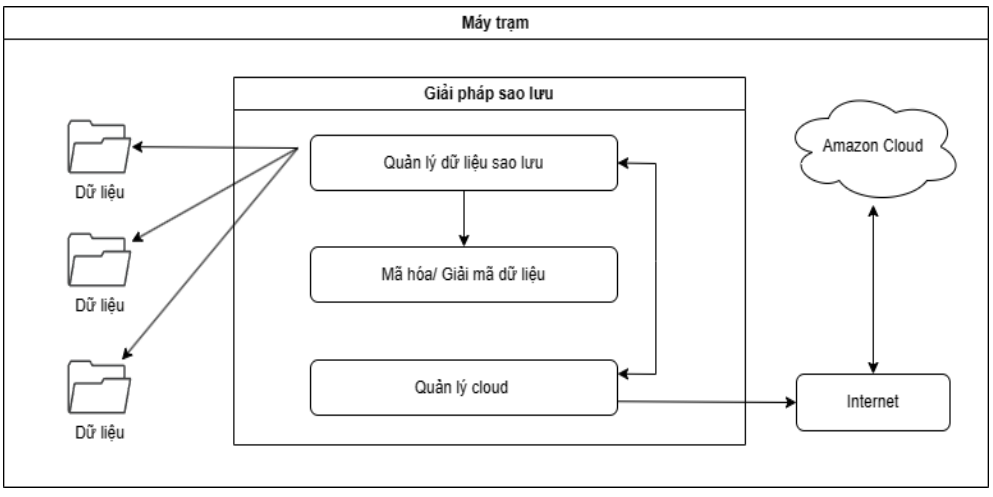
  Cần xác định sao lưu các file cụ thể, loại dữ liệu cụ thể trên máy trạm hay sao lưu theo thư mục

- Bước 2: Lựa chọn cơ chế sao lưu

  Thực hiện sao lưu toàn bộ dữ liệu hay chỉ sao lưu các dữ liệu có sự thay đổi

- Bước 3: Thực hiện sao lưu, mã hóa dữ liệu sao lưu
- Bước 4: Kết nối với server Cloud lưu trữ
- Bước 5: Khôi phục lại dữ liệu từ server Cloud lưu trữ.

Với các phân tích như trên, mô hình tổng quát của hệ thống sẽ bao gồm các thành phần:



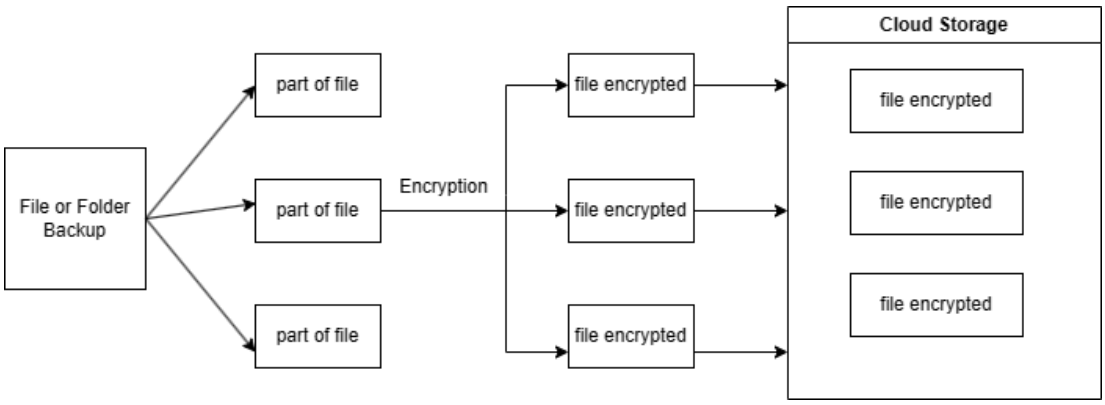

*Hình 1.5 Mô hình và giải pháp tổng thể*

- Module quản lý dữ liệu sao lưu: Quản lý các file cần sao lưu và lên kế hoạch sao lưu cho các dữ liệu này. Module này cũng quản lý việc khôi phục các file khi có yêu cầu.

- Module mã hóa/ giải mã dữ liệu: Phụ trách việc mã hóa các file dữ liệu trước khi chúng được đưa lên cloud. Khi khôi phục dữ liệu, module này sẽ giải mã các dữ liệu đã mã hóa để khôi phục lại bản rõ ban đầu.
- Module S3: Quản lý việc kết nối, xác thực và trao đổi dữ liệu với các nền tảng lưu trữ đám mây.

Giải pháp sao lưu dữ liệu bao gồm 2 luồng nghiệp vụ chính: Sao lưu dữ liệu lên đám mây lưu trữ và khôi phục dữ liệu từ đám mây lưu trữ về máy. Mô hình đề xuất sẽ hoạt động hoàn toàn dưới máy trạm mà không cần thông qua một server trung gian nào. Giải pháp này có thể tiết kiệm chi phí khi không phải duy trì chi phí cho server, domain và SSL, mặt khác quá trình quản lý khóa và mã hóa file nằm hoàn toàn trên máy trạm, tránh việc attacker có thể đứng giữa nghe lén quá trình trao đổi dữ liệu.

**Quá trình sao lưu:**

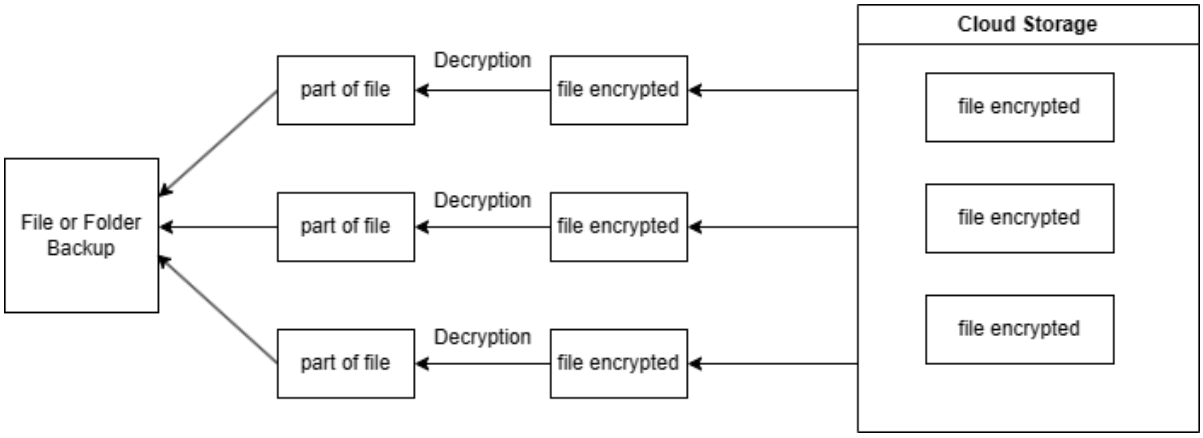



*Hình 1.6 Quá trình sao lưu dữ liệu*

Quá trình sao lưu dữ liệu được chia thành các bước chính:

- File Backup được chia nhỏ thành các phần, việc chia nhỏ giúp quá trình mã hóa được thực hiện song song, quá trình upload file lên server backup cũng nhanh hơn. Mặt khác với các dịch vụ backup được thiết kế tốt, khi file có thay đổi thì chỉ phần nào có thay đổi thì mới cần đẩy lên server.
- Mã hóa các phần của file đã được chia nhỏ. Thường các hệ thống sẽ sử dụng các kỹ thuật mã hóa đối xứng để mã hóa ở bước này.
- Đẩy các file đã được mã hóa lên server Backup.

**Quá trình khôi phục dữ liệu:**




*Hình 1.7 Quá trình khôi phục dữ liệu*

Quá trình khôi phục dữ liệu được chia thành các bước chính:

- Lấy các phần của file đã mã hóa từ server cloud backup về local.
- Giải mã các thành phần của file.
- Ghép các phần của file để được file hoàn chỉnh.

Trong mô hình mô tả trên việc phân tách các tệp lớn thành các thành phần nhỏ hơn trước khi thực hiện mã hóa. Điều này không chỉ cho phép mã hóa hiệu quả từng phần riêng lẻ mà còn mang lại nhiều lợi ích đáng kể:

**Tăng cường khả năng xử lý song song**: Việc chia nhỏ tệp thành các phần cho phép hệ thống sử dụng luồng độc lập để mã hóa, tăng tốc độ xử lý đáng kể.

**Cải thiện hiệu quả tiến trình tải lên và tải xuống**: Khi một phần của tệp gặp sự cố trong quá trình tải lên, chỉ cần tải lại phần đó thay vì toàn bộ tệp, giảm thiểu đáng kể thời gian và băng thông cần thiết.

**Hiệu quả cao trong quản lý không gian lưu trữ**: Khi chỉ có một phần của tệp thay đổi, chỉ cần cập nhật phần đó. Những phần không thay đổi không cần phải tải lên lại, tiết kiệm không gian lưu trữ và giảm chi phí.

### 1.2.2 Mã hóa bất đối xứng RSA

RSA là một thuật toán mã hóa công khai, tên của thuật toán được tạo ra từ ba nhà toán học người Anh: Ron Rivest, Adi Shamir và Leonard Adleman. Thuật toán RSA hoạt động dựa trên sự khó khăn trong việc phân tích một số nguyên

lớn thành các thành phần nguyên tố của nó. Khóa bí mật và khóa công khai của RSA là hai số nguyên tố lớn, và bí mật nằm ở việc tìm ra các số nguyên tố này từ tích của chúng.

Thuật toán RSA có thể mô tả như sau:

- Khởi tạo một cặp khóa bao gồm khóa công khai và khóa bí mật. Khóa công khai sẽ được chia sẻ với mọi người, khóa bí mật sẽ được giữ kín.
- Chọn hai số nguyên tố lớn p và q với p # q, lựa chọn ngẫu nhiên và độc lập và sau đó tính modulus n = q * p
- Tính giá trị hàm số Euler φ(n) = (p - 1)(q - 1).
- Chọn một số nguyên e sao cho 1 < e < φ(n) và e là số nguyên tố cùng nhau với φ(n).
- Tìm d sao cho d * e ≡ 1 (mod φ(n)). Điều này có nghĩa là d * e chia hết cho φ(n) và có số dư là 1 khi chia cho φ(n).
- Khóa công khai bao gồm: n(modulus) và e(số mũ công khai)
- Khóa bí mật bao gồm: n(modulus) và d ( số mũ bí mật) ngoài ra còn một dạng khác của khóa bí mật bao gồm:

  + q và p hai số nguyên tố chọn ban đầu
  + d mod (p-1) và d mod (q-1) được gọi là dmp1 và dmq1
  + (1/q) mod p được gọi là iqmp

Mã hóa: một thông điệp được biểu diễn dưới dạng số nguyên m. Sử dụng khóa công khai e, tính c = m^e (mod n).

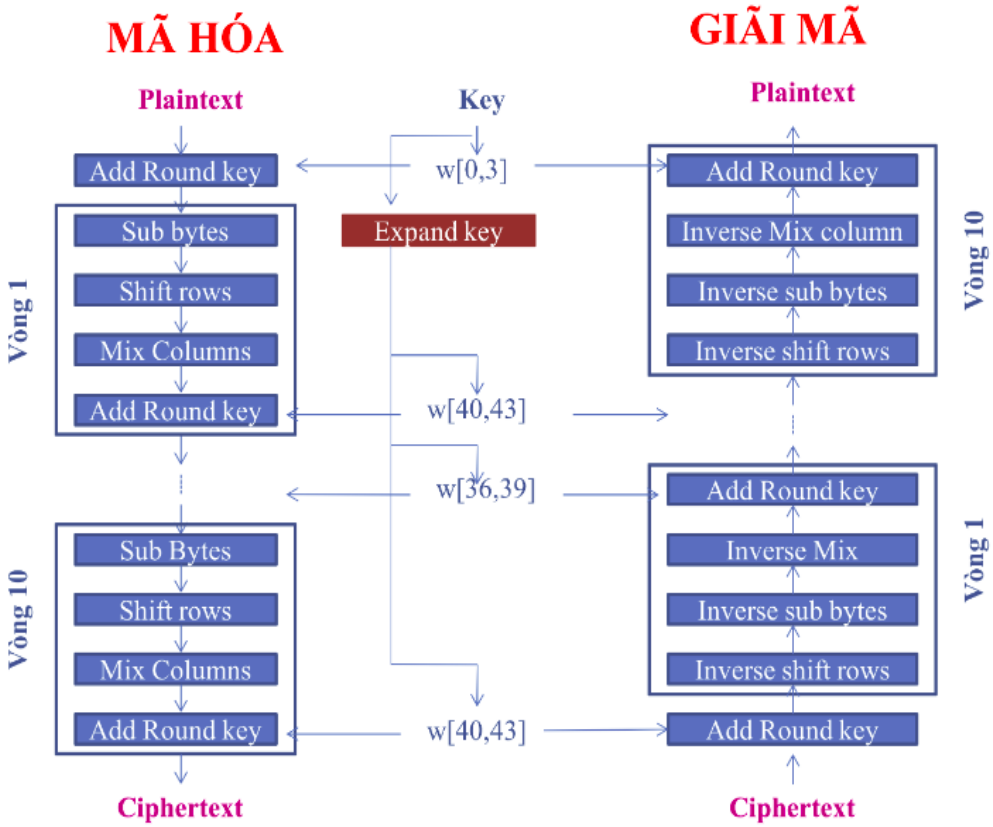
Giải mã: Sử dụng khóa bí mật d, tính toán m = c^d (mod n).

### 1.2.3 Mã hóa đối xứng AES

AES (viết tắt của Advanced Encryption Standard ) là một thuật toán mã hoá khối được thiết kế bởi Rijndael. Thuật toán hoạt động dựa trên một loạt phép biến đổi đối xứng, bao gồm các vòng lặp và các hộp S-box, cùng với một phép trộn các hàng và cột. Quá trình bắt đầu bằng việc chia khối dữ liệu thành các khối nhỏ hơn, thường là 128 bit. Mỗi khối này sẽ trải qua một loạt các

vòng biến đổi, các khóa con sử dụng trong các chu trình được tạo ra bởi quá trình tạo khóa con Rijndael.

Số vòng lặp (kí hiệu là $N_r$ ) phụ thuộc vào độ dài khóa, nếu độ dài khóa là 256 bit thì $N_r = 14$




*Hình 1.8 Mô hình mã hóa và giải mã bằng thuật toán AES*

Thuật toán AES tổng quát có thể mô tả như sau:

Định nghĩa một văn bản cho trước x được biểu diễn dưới dạng một ma trận 4x4 được gọi là state. Đầu tiên thực hiện phép XOR giữa state và một khóa vòng (RoundKey) gọi là Add Round Key.

Sau đó cho mỗi vòng lặp quá trình mã hóa:

- Thực hiện phép thay thế byte trong state bằng cách sử dụng một SBox, gọi là SubBytes.
- Dich chuyển các hàng của state theo một cách xác định gọi là ShiftRows.
- Áp dụng một phép biến đổi tuyến tính trên mỗi cột của state, gọi là Mix Cloumns

Tiếp tục thực hiện phép XOR giữa state và một khóa vòng. Trong vòng lặp cuối cùng:

- Thực hiện các phép dịch chuyển hàng, ShiftRows.
- Áp dụng phép thay thế byte cuối cùng trong state bằng một SBox
- Cuối cùng thực hiện phép XOR giữa state và khóa vòng

Kết quả là văn bản mã, được biểu diễn bằng state sau khi hoàn thành tất cả các vòng lặp.

### 1.2.4 Lược đồ mã hóa đầu cuối

Mã hóa bất đối xứng RSA dựa trên nguyên tắc toán học của số nguyên lớn và phép toán trên các số nguyên rất lớn. Phép toán chính trong RSA là lũy thừa mô-đun, một phép toán tốn kém về mặt tính toán, đặc biệt khi các số nguyên lớn làm tăng đáng kể độ phức tạp.

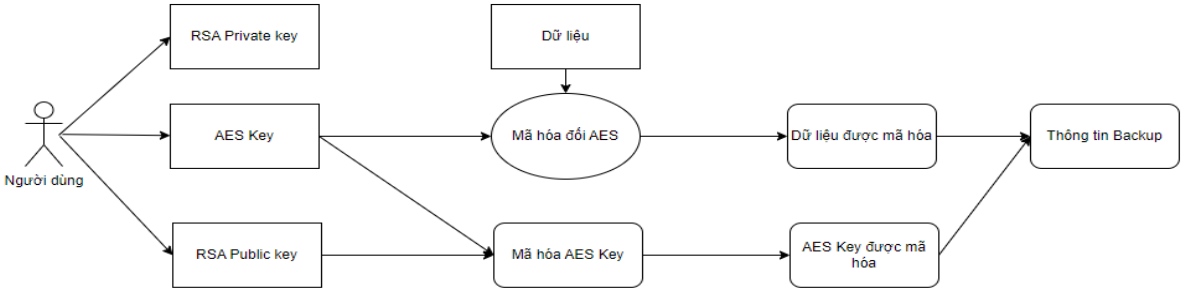
Mã hóa đối xứng AES dựa trên nguyên tắc xử lý dữ liệu theo khối. AES sử dụng một chuỗi các phép biến đổi thay thế và linh hoạt và ít tốn kém hơn về mặt tính toán so với phép lũy thừa mô đun của RSA.

Việc xử lý dữ liệu theo khối và sử dụng các phép toán đơn giản hơn làm cho AES phù hợp cho việc mã hóa dữ liệu lớn.

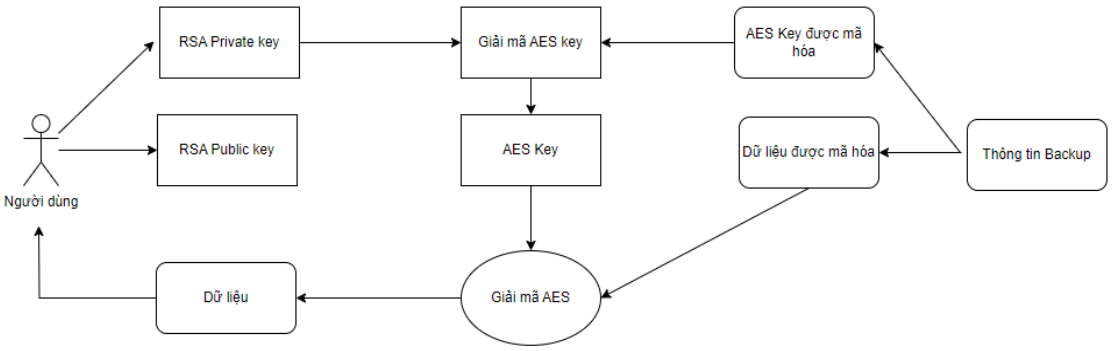
Sau khi làm rõ về mã hóa và các kiểu mã hóa đang được sử dụng nhiều hiện nay.Có thể lựa chọn được giải thuật mã hóa sẽ sử dụng như sau:

- Sử dụng mã hóa đối xứng AES để mã hóa các dữ liệu backup
- Sử dụng mã hóa bất đối xứng RSA để mã hóa key của thuật toán AES.

Với việc sử dụng hai thuật toán trên, sản phẩm xây dựng cũng đã đạt độ an toàn bảo mật cao, đảm bảo hiệu năng và tính khả dụng khi sử dụng thực tế.
Sau đây là mô hình hóa cho thiết kế đã phân tích nêu trên.



*Hình 1.9 Mô hình mã hóa file*




*Hình 1.10 Mô hình giải mã file*

Nếu chỉ sử dụng giải pháp mã hóa AES mà không sử dụng RSA thì bắt buộc giải pháp phải sử dụng một một khóa AES cho tất cả các bản backup của file. Và khóa AES cần được lưu trữ trong database. Việc người dùng cho một khóa AES cho nhiều cấu hình backup khác nhau cũng dễ xảy ra, khóa AES này cũng thường không có độ phức tạp đủ tốt khi người dùng tự đặt.

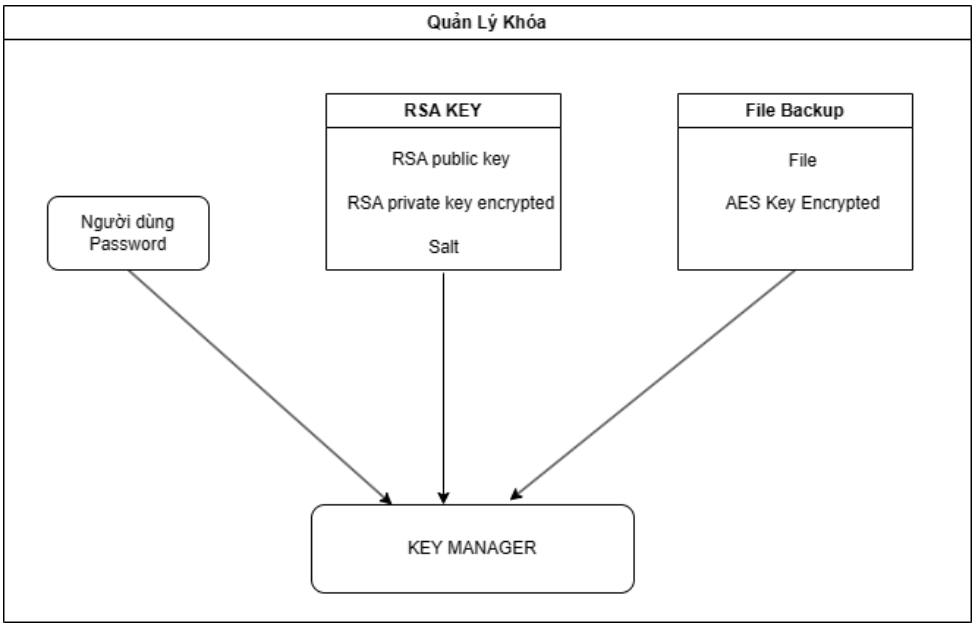
Khi sử dụng kết hợp RSA, khóa AES có được được hệ thống sinh ngẫu nhiên với độ phức tạp cao. Dữ liệu mã hóa có thể được gắn theo khóa AES đã được mã hóa. Do đó với mỗi bản backup hoàn toàn có thể sử dụng một khóa AES sinh ngẫu nhiên khác nhau.

Với cách tiếp cận như vậy, rất khó để có thể lần ra được dữ liệu gốc từ dữ liệu backup.

### 1.2.5 Quản lý và lưu trữ các khóa

Có 3 khóa cần lưu trữ trong hệ thống:

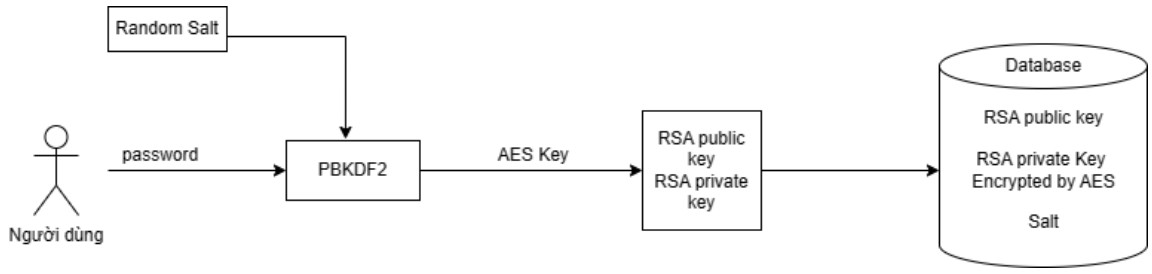
- Khóa của thuật toán AES
- Khóa công khai của thuật toán RSA
- Khóa bí mật của thuật toán RSA




*Hình 1.11 Mô hình quản lý và lưu trữ các khóa*

Vì người dùng chỉ có duy nhất thông tin là mật khẩu đăng nhập hệ thống do đó sẽ sử dụng mật khẩu này để dẫn xuất ra AES key phục vụ mã hóa RSA Private key. Như vậy thì không cần phải lưu trữ AES key này vào hệ thống và mật khẩu của người dùng chính là chìa khóa để giải mã được tất cả các thông tin này.

Từ đấy thiết kế được một sơ đồ logic như sau:




*Hình 1.12 Sơ đồ logic quản lý khóa*

Sử dụng thuật toán mã hóa khác để sinh ra AES key từ mật khẩu người dùng, PBKDF2 (Password-Based Key Derivation Function 2) là một chức năng dẫn xuất khóa dựa trên mật khẩu, được thiết kế để chuyển đổi mật khẩu không an toàn thành một khóa bí mật hoặc một chuỗi ngẫu nhiên dài hơn và an toàn hơn. Được chuẩn hóa trong RFC 2898 (cập nhật bởi RFC 8018), PBKDF2

được sử dụng rộng rãi để tăng cường bảo mật của mật khẩu bằng cách làm cho việc tấn công mật khẩu trở nên khó khăn hơn.

PBKDF2 hoạt động theo bốn bước chính:

- **Nhập liệu**: PBKDF2 cần nhập liệu là mật khẩu của người dùng, một giá trị salt (một chuỗi ngẫu nhiên được thêm vào để làm tăng tính duy nhất và giảm sự trùng lặp), và số lần lặp (iteration count). Số lần lặp càng cao, quá trình sinh khóa càng an toàn nhưng đòi hỏi nhiều tài nguyên tính toán hơn
- **Hàm băm**: PBKDF2 sử dụng một hàm băm mật mã, thường là SHA-1, SHA-256 hoặc SHA-512. Hàm băm này sẽ xử lý các dữ liệu đầu vào và tạo ra một chuỗi kết quả
- **Tạo khóa**: PBKDF2 áp dụng hàm băm lên mật khẩu đã kết hợp với salt trong nhiều vòng lặp. Mỗi lần lặp lại là một quá trình băm mới, và kết quả của mỗi lần băm được tích lũy hoặc kết hợp theo cách nhất định để tăng độ phức tạp và bảo mật
- **Xuất khóa**: Kết quả cuối cùng của quá trình băm lặp là khóa hoặc chuỗi đã được dẫn xuất. Khóa này có thể dài hơn nhiều so với mật khẩu gốc và khó bị phá vỡ hơn nhiều.

AES key này được sinh trực tiếp từ mật khẩu người dùng thông qua PBKDF2 do đó không cần được lưu trữ lại trong hệ thống.

Khi một cấu hình backup được tạo, một AES key sẽ được sinh ngẫu nhiên và trong suốt với người dùng. Module quản lý khóa sẽ sử dụng RSA public key để mã hóa key AES lại và lưu key AES đã mã hóa vào database cùng với các thông tin khác của cấu hình AES vừa tạo.

Khi cần sử dụng tới key AES thì người dùng sẽ cung cấp mật khẩu và hệ thống sẽ chạy lại logic quản lý khóa để module quản lý khóa lại giải mã và cung cấp cho module cần dùng tới.

## 1.3. Giải pháp và công nghệ sử dụng

### 1.3.1 Giao thức S3

#### a) Giới thiệu

S3 (Simple Storage Service) là một giao thức được sử dụng khá phổ biến, được cung cấp bởi Amazon Web Services. S3 cung cấp một giao diện dịch vụ web RESTful dựa trên HTTP/HTTPS, cho phép truyền dữ liệu một các liền mạch giữa cloud và máy khách. Điều này không những tạo điều kiện thuận lợi cho việc tích hợp với các ứng dụng mà còn đảm bảo tính năng động và dễ dàng mở rộng.

Với S3, người dùng có thể lưu trữ mọi loại từ hình ảnh, video, tài liệu cho đến các loại dữ liệu phi cấu trúc khác. Điểm nổi bật của S3 không chỉ là khả năng lưu trữ dữ liệu mà còn bao gồm việc truy xuất lịch sử các phiên bản của dữ liệu, kiểm soát quyền truy cập một các minh bạch và thiết lập các chính sách vòng đời cho dữ liệu. Tính năng này không chỉ tăng cường độ bảo mật và tuân thủ các quy định mà còn giúp tối ưu hóa chi phí lưu trữ.

S3 là một giải pháp lưu trữ đáng tin cậy, bảo mật và chi phí hiệu quả, làm cho nó trở thành lựa chọn ưu tiên cho cả doanh nghiệp và cá nhân mong muốn khai thác lợi ích của việc lưu trên đám mây. Với khả năng mở rộng linh hoạt, S3 cung cấp nền tảng vững chắc cho việc lưu trữ và quản lý dữ liệu lớn, phục vụ cho nhiều mục đích từ phân tích big data đến sao lưu và phục hồi thảm họa.

**b) Dịch vụ Amazon S3**

Amazon Simple Storage Service (Amazon S3) là một dịch vụ lưu trữ đối tượng đa năng và mạnh mẽ của Amazon Web Services, được thiết kế để cung cấp độ bảo mật, khả năng mở rộng, và hiệu suất cao. S3 cho phép người dùng lưu trữ và bảo vệ mọi loại dữ liệu, lưu trữ dữ liệu dài hạn, ứng dụng doanh nghiệp, thiết bị IOT, và phân tích dữ liệu lớn.

Kiến trúc của Amazon S3 bao gồm các thành phần chính sau:

- **Regions**: Amazon S3 có sẵn tại nhiều khu vực trên toàn cầu, mỗi khu vực đại diện cho một vị trí địa lý cụ thể. Người dùng có thể chọn khu vực lưu trữ phù hợp để tối ưu hóa hiệu suất và tuân thủ các quy định về bảo vệ dữ liệu.

- **Object Storage**: Dữ liệu trong S3 được lưu trữ dưới dạng đối tượng, mỗi đối tượng được xác định bởi một khóa duy nhất. Đối tượng có thể là bất kỳ loại file nào và được lưu trữ dưới dạng không thay đổi, đảm bảo tính khả dụng và độ tin cậy cao.
- **Buckets**: Đối tượng được lưu trữ trong các "bucket", tương tự như thư mục trong hệ thống file, với mỗi bucket có một tên định danh toàn cầu duy nhất và gắn với một khu vực địa lý cụ thể.
- **RESTful API**: Amazon S3 sử dụng giao diện lập trình ứng dụng (API) dựa trên REST, cho phép người dùng lưu trữ và truy xuất dữ liệu thông qua giao thức HTTP/HTTPS.
- **Security**: S3 cung cấp các lớp bảo mật nhiều tầng bao gồm quản lý danh tính và truy cập, mã hóa dữ liệu và kiểm soát quyền truy cập qua các chính sách do người dùng xác định.
- **Lifecycle Policies**: Chính sách vòng đời của S3 cho phép người dùng tự động xóa hoặc di chuyển dữ liệu dựa trên thời gian tồn tại, giúp quản lý dữ liệu một các hiệu quả.

Với những tính năng nổi bật này, Amazon S3 là một giải pháp lưu trữ đáng tin cậy và linh hoạt, đáp ứng nhu cầu lưu trữ và quản lý dữ liệu của doanh nghiệp lớn và cá nhân, trên quy mô toàn cầu.

### 1.3.2 Duplicati

Duplicati là một ứng dụng sao lưu mã nguồn mở, hỗ trợ nhiều nền tảng, bao gồm Windows, macOS và Linux. Duplicati không có tác động của các thành phần máy chủ và do đó, nó có thể hỗ trợ nhiều nhà cung cấp dịch vụ lưu trữ dựa trên đám mây. Duplicati lưu trữ theo cách thức chia nhỏ các file lớn, khi phục hồi dữ liệu thì hợp nhất các file nhỏ đó lại, và hỗ trợ các tính năng như mã hoá, nén và loại bỏ trùng lặp, quản lý phiên bản và sao lưu tăng tiến (incremental backup). Các dịch vụ lưu trữ sao lưu mà Duplicati hỗ trợ là rất đa dạng, có thể kể đến Google Drive, Dropbox, Amazon S3, FTP, WebDAV… Hệ

thống Duplicati sử dụng chuẩn mã hoá AES-256 để mã hoá đầu cuối và hệ quản trị cơ sở dữ liệu là SQLite .

Duplicati là một phần mềm sao lưu và phục hồi dữ liệu mạnh mẽ và linh hoạt, tuy nhiên, nó cũng có nhược điểm liên quan đến độ phức tạp và hiệu suất.

Do Duplicati có thể yêu cầu tài nguyên máy tính và thời gian để thực hiện quá trình sao lưu và phục hồi dữ liệu, bao gồm nhiều bước đòi hỏi cường độ tính toán cao, điều này có thể ảnh hưởng đến hiệu suất chung của hệ thống.

**a) Quá trình sao lưu**

Giả sử muốn backup một thư mục trên máy tính Window, nội dung của thư mục như sau:

C:\data

|--------> mydoc.txt, 4kb

|--------> mydiveo.mp4, 2Mb

|--------> extra

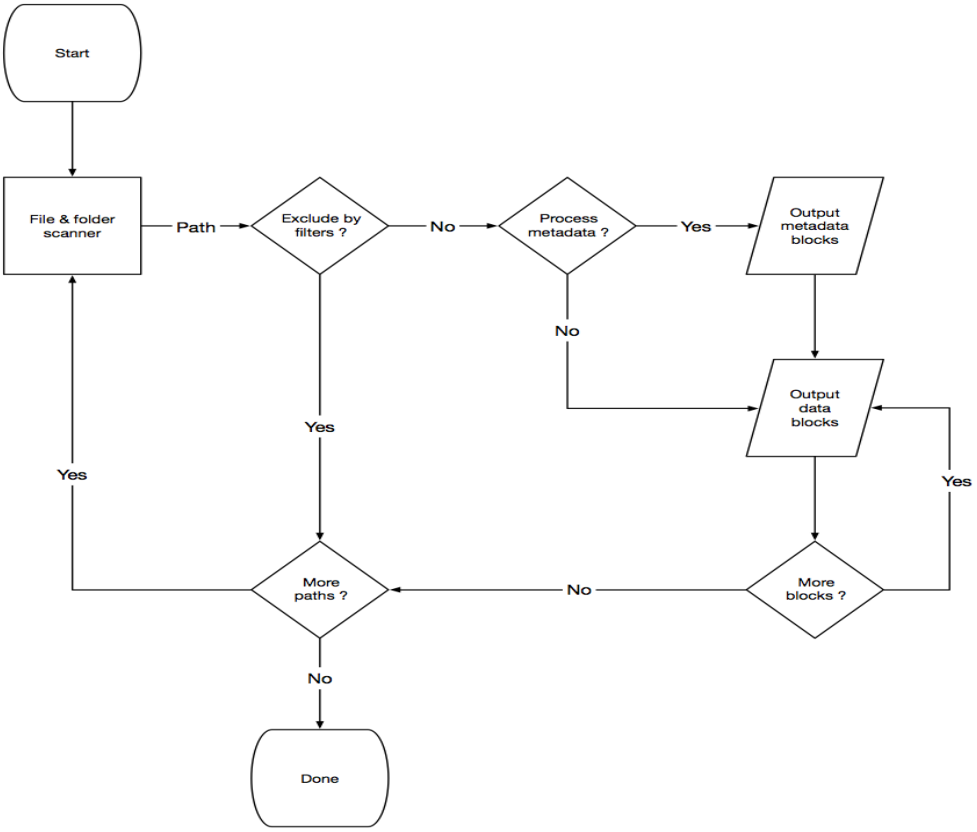
|--------> olddoc.txt, 2kb

|--------> samevideo.mp4, 210kb

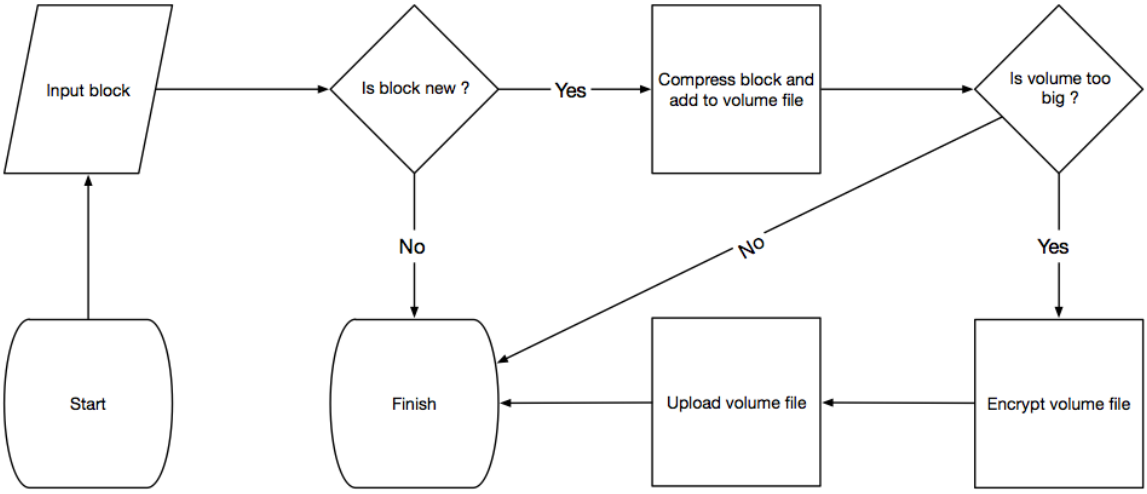
Để lưu trữ thông tin về những gì có trong bản sao lưu, Duplicati dựa vào các định dạng tệp tiêu chuẩn và sử dụng định dạng dữ liệu JSON và định dạng file nén zip.

Danh sách các tên file sẽ được Duplicati tạo một tệp tin có tên là *duplicati-2024101409000.dblist.zi* trong đó các số biểu thị ngày và giờ hiện tại theo múi giờ UTC. Bên trong file zip này là một tệp JSON có tên là *filelist.json*, bắt đầu bằng một danh sách trống, được biểu thị bằng JSON dưới dạng [].

Dữ liệu của file thì sẽ được Duplicati tạo một tệp có tên là *7af781d3401eb90cd371.dblock.zip* ở đây tên file được tạo ra ngẫu nhiên và không liên quan đến ngày giờ hệ thống. Sau đây là mô hình quá trình xử lý sao lưu file.




*Hình 1.13 Quá trình thực hiện sao lưu của Duplicati*




*Hình 1.14 Quá trình xử lý khối dữ liệu*

Quá trình xử lý các khối dữ liệu:

Ở bước đầu tiên, Duplicati sẽ duyệt qua hệ thống các thư mục và file, từ đó cho ra đường dẫn tuyệt đối cho các thư mục và file cần sao lưu:

```
C:\data\
C:\data\mydoc.txt
C:\data\myvideo.mp4
C:\data\extra\
C:\data\extra\olddoc.txt
C:\data\extra\samevideo.mp4
```

Khi bắt đầu sao lưu, Duplicati đọc đối tượng đầu tiên, *C:\data*. Đối tượng này là một thư mục, do đó hệ thống chỉ thêm các thông tin của đối tượng này vào file *filelist.json*:

```
[
  {
    "type": "folder",
    "path": "C:\\data\\"
  }
]
```

Duplicati sẽ đọc file đó theo từng khối có kích thước theo cài đặt hoặc mặc định là 100KB. Đối tượng tiếp theo là một file "*C:\data\mydoc.txt*." có kích thước là 4kb bé hơn kích thước của một khối mặc định, nên toàn bộ nội dung tập sẽ vừa vặn trong một khối duy nhất. Sau khi đọc tệp, Duplicati tạo một hàm băm SHA-256 cho khối dữ liệu này và sau encode theo định dạng Base64, kết quả cuối cùng có dạng

```
qaFXpxVTuYCuibb9P41VSeVn4pIaK8o3jUpJKqI4VF4=.
```

Nội dung của tệp mydoc.txt (4kb) sau đó được thêm vào tệp dblock được đề cập ở trên, sử dụng chuỗi làm tên tệp. Điều này có nghĩa là nội dung tệp dblock zip giờ đây là:

```
[
  {
    "type": "Folder",
    "path": "C:\\data\\"
  },{
    "type": "File",
    "path": "C:\\data\\mydoc.txt",
    "size": 4096,
    "hash":
   "qaFXpxVTuYCuibb9P41VSeVn4pIaK8o3jUpJKqI4VF4="
  }
]
```

Sau đó, với đối tượng *C:\data\myvideo.mp4*, đây là một file có kích thước lớn. Phương pháp xử lý với file này cũng tương tự như file *C:\data\mydoc.txt*. Tuy nhiên do kích thước file này lớn hơn kích thước của một khối (210KB so với 100KB), Duplicati sẽ đọc nó thành ba khối và tính toàn hàm băm SHA-256 cho ba khối đó, hai khối đầu tiên có kích thước 100KB và khối còn lại là 10KB. Từng khối dữ liệu được thêm vào file dblock, lúc này dữ liệu trong file có dạng:

```
qaFXpxVTuYCuibb9P41VSeVn4pIaK8o3jUpJKqI4VF4= (4kb)
0td8NEaS7SMrQc5Gs0Sdxjb/1MXEEuwkyxRpguDiWsY= (100kb)
PN2oO6eQudCRSdx3zgk6SJvlI5BquP6djt5hG4ZfRCQ= (100kb)
uS/2KMSmm2IWlZ77JiHH1p/yp7Cvhr8CKmRHJNMRqwA= (10kb)
```

Thêm vào đó, hệ thống sẽ tính hàm băm cho toàn bộ file và thu được chuỗi dữ liệu dạng:

```
4sGwVN/QuWHD+yVI10qgYa4e2F5M4zXLKBQaf1rtTCs=
```

Bây giờ *filelist.json* lưu trữ như sau:

```
{
  "type": "File",
  "path": "C:\\data\\myvideo.mp4",
  "size": 215040,
```

```
        "hash":
"4sGwVN/QuWHD+yVI10qgYa4e2F5M4zXLKBQaf1rtTCs=",
        "blocks": [
          "0td8NEaS7SMrQc5Gs0Sdxjb/1MXEEuwkyxRpguDiWsY=",
          "PN2oO6eQudCRSdx3zgk6SJvlI5BquP6djt5hG4ZfRCQ=",
          "uS/2KMSmm2IWlZ77JiHH1p/yp7Cvhr8CKmRHJNMRqwA="
        ]
      }
```

Tuy vậy, có một vấn đề là ta sẽ lưu trữ khoảng 47 ký tự cho mỗi 100kb dữ liệu, nên tệp 1GB sẽ thêm 482kb dữ liệu bổ sung vào danh sách tệp làm cho hệ thống danh sách tệp trở nên cực kỳ lớn. Thay vào đó, Duplicati tạo thêm một "khối gián tiếp", một khối dữ liệu mới chỉ với các giá trị băm. Vì dữ liệu đầu ra của hàm băm SHA-256 có độ dài là 32 byte nên nếu không được encode bằng base64, ta có thể lưu trữ 3200 khối dữ liệu băm (kích thước khoảng 300MB) trong một khối duy nhất, nghĩa là kích thước file *filelist.json* sẽ chỉ tăng thêm 47 byte cho 300MB dữ liệu.

Đối với file *C:\data\myvideo.mp4*, hệ thống tạo ra ba khối, vì vậy khối mới chứa ba chuỗi khối băm chỉ chiếm 96 byte. Khối mới này được xử lý không khác gì các khối khác và hàm băm SHA-256 được tính toán, kết quả sau khi được mã hoá base64 có dạng:

```
Uo1f4rVjNRX10HkxQxXauCrRv0wJOvStqt9gaUT0uPA=
```

Đối tượng mới được thêm vào file *filelist.json*:

```
      [
        {
        "type": "Folder",
        "path": "C:\\data\\"
        },{
        "type": "File",
        "path": "C:\\data\\mydoc.txt",
        "size": 4096,
        "hash": "qaFXpxVTuYCuibb9P41VSeVn4pIaK8o3jUpJKqI4VF4="
        }, {
```

```
        "type": "File",
        "path": "C:\\data\\myvideo.mp4",
        "size": 215040,
        "hash":
"4sGwVN/QuWHD+yVI10qgYa4e2F5M4zXLKBQaf1rtTCs=",
        "blocklists":                                    [
"Uo1f4rVjNRX10HkxQxXauCrRv0wJOvStqt9gaUT0uPA=" ]
        }
      ]
```

Các file có kích thước lớn hơn một khối khác cũng được Duplicati xử lý tương tự, và kết quả sau cùng file filelist.json có dạng như sau:

```
      [
        {
          "type": "Folder",
          "path": "C:\\data\\"
        },
        {
          "type": "File",
          "path": "C:\\data\\mydoc.txt",
          "size": 4096,
          "hash":
"qaFXpxVTuYCuibb9P41VSeVn4pIaK8o3jUpJKqI4VF4="
        },
        {
          "type": "File",
          "path": "C:\\data\\myvideo.mp4",
          "size": 215040,
          "hash":
"4sGwVN/QuWHD+yVI10qgYa4e2F5M4zXLKBQaf1rtTCs=",
          "blocklists": [
            "Uo1f4rVjNRX10HkxQxXauCrRv0wJOvStqt9gaUT0uPA="
          ]
        },
        {
```

```
            "type": "File",
            "path": "C:\\data\\extra\\olddoc.txt",
            "size": 2048,
            "hash":
  "R/XSNsb4ln/SkeJwFDd4Fv4OnW2QNIxMR4HItgg9qCE="
          },
          {
            "type": "File",
            "path": "C:\\data\\extra\\samevideo.mp4",
            "size": 215040,
            "hash":
  "4sGwVN/QuWHD+yVI10qgYa4e2F5M4zXLKBQaf1rtTCs=",
            "blocklists": [
              "Uo1f4rVjNRX10HkxQxXauCrRv0wJOvStqt9gaUT0uPA="
            ]
          },
        ]
```


**b) Quá trình khôi phục**

Tiếp tục với ví dụ trên, quá trình khôi phục dữ liệu của Duplicati sử dụng file filelist.json, ở đây ta cần khôi phục 4 file và có file cần blocklist. Do đó, quá trình khôi phục bắt đầu với việc trích xuất blocklist thành các khối băm cần thiết. Vì trong danh sách file có hai file có cùng blocklist, nên ta chỉ cần lấy dữ liệu từ khối này. Tên của các file dblock và dữ liệu chúng chứa không có liên hệ nào, vì vậy ta cần tải xuống tất cả các file cho đến khi tìm thấy dữ liệu cần dùng. Điều này sẽ làm giảm hiệu năng của ứng dụng trong thực tế, nên Duplicati sẽ ghi các thông tin về các dblock này chứa những khối băm nào trong các file dindex.

Tiếp theo, ta có thể dựa vào kích thước file blocklist hoặc kích thước file thực tế (trong filelist.json) để tính ra số lượng khối hash cần lấy, ở đây là ba khối có biểu diễn dưới dạng base64 là:

```
0td8NEaS7SMrQc5Gs0Sdxjb/1MXEEuwkyxRpguDiWsY= (100kb)
PN2oO6eQudCRSdx3zgk6SJvlI5BquP6djt5hG4ZfRCQ= (100kb)
uS/2KMSmm2IWlZ77JiHH1p/yp7Cvhr8CKmRHJNMRqwA= (10kb)
```

Đối với các file có kích thước nhỏ, việc khôi phục được thực hiện không quá phức tạp: trích xuất dữ liệu và lưu vào file có tên tương ứng. Quá trình xác định vị trí file dblock chứa khối dữ liệu mà ta cần và giải nén được cải thiện và đơn giản hoá trong Duplicati với các file dindex.

Đối với các file có kích thước lớn, khi đã có danh sách khối băm cần dùng như trên, ta sẽ tiến hành khôi phục từng khối. Việc khôi phục có thể tiến hành theo thứ tự từng khối băm một, giải nén từng khối và thêm dữ liệu lần lượt vào file đích, hoặc khôi phục không theo thứ tự (do ta đã biết sẵn kích thước một khối), lúc này ta có thể tính độ dời của dữ liệu trước rồi thêm dữ liệu vào đúng vị trí trong file đích.

Sau khi các file được khôi phục, hệ thống sẽ tính toán giá trị băm của từng file và so sánh với giá trị được lưu trong file filelist.json, nếu chúng giống nhau thì hệ thống xác nhận việc khôi phục dữ liệu thành công.

## 1.4. Tổng kết chương

Trong chương này, đồ án đã trình bày tổng quan về đề tài: “Xây dựng hệ thống sao lưu tự động sử dụng giao thức S3 có tích hợp mã hóa đầu cuối ” trên các khía cạnh: nắm bắt bối cảnh và tầm quan trọng của nghiên cứu, xác định mục tiêu và phạm vi của đề tài, bên cạnh đó đồ án đã giới thiệu các chiến lược sao lưu dữ liệu, mã hóa đầu cuối, bài toán cần giải quyết, cũng như giải pháp và công nghệ sử dụng trong quá trình nghiên cứu.

Giải pháp sao lưu dữ liệu bao gồm 2 luồng nghiệp vụ chính: sao lưu và mã hóa dữ liệu đẩy lên đám mây và lưu trữ, giải mã và khôi phục dữ liệu từ đám mây về máy. Mô hình đề xuất hoạt động máy trạm mà không thông qua một server trung gian nào. Giải pháp này có thể tiết kiệm chi phí và mặt khác có thể tránh việc bị tấn công nghe lén trong quá trình trao đổi dữ liệu.

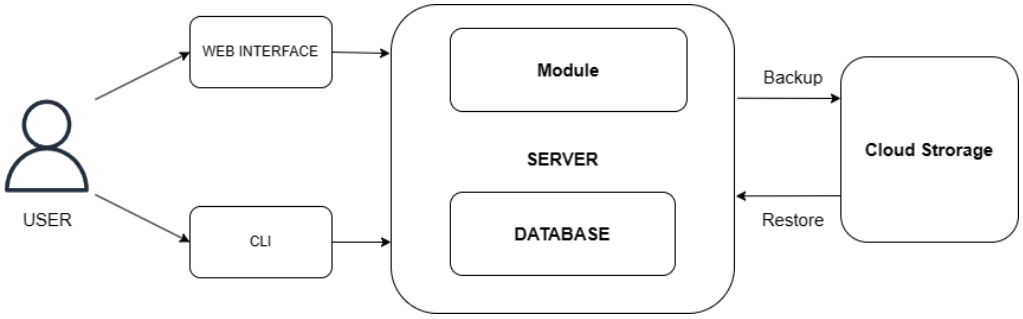
Để làm được điều đó, đồ án đề xuất sử dụng giao thức S3 của Amazon Web Services (AWS) làm nền tảng để xây dựng hệ thống sao lưu và phục hồi

dữ liệu, cùng với công nghệ hỗ trợ của Duplicati bao gồm các API có chức năng sao lưu và phục hồi dữ liệu. Điều này có thể giúp tạo ra một hệ thống hoàn chỉnh hơn.

Trên cơ sở tổng quan về chiến lược sao lưu, bài toán, giải pháp và công nghệ sử dụng, đồ án đã xác định hướng đi của nghiên cứu và trong chương tiếp theo sẽ tiến hành phân tích, thiết kế một hệ thống sao lưu và phục hồi dữ liệu tích hợp mã hóa đầu cuối dựa trên nền tảng của chương này.

# CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1. Tổng quan về hệ thống




*Hình 2.1 Sơ đồ tổng quan hệ thống*

Hệ thống bao gồm các thành phần chính như sau:

**Giao diện quản trị**

Người dùng có thể lựa chọn một trong hai cách để giao tiếp với hệ thống, qua giao diện web hoặc giao diện dòng lệnh để cấu hình hệ thống, cấu hình các task backup, chạy backup, chạy restore

**Backend**

Đây là thành phần trung tâm tương tác với các nhà cung cấp dịch vụ lưu trữ, lưu lại các dữ liệu hệ thống trong một cơ sở dữ liệu được nhúng trực tiếp vào ứng dụng (SQLite). Thực hiện các chức năng đăng nhập, đăng xuất, cấu hình các task backup, xác thực với dịch vụ cloud bên ngoài, khôi phục dữ liệu backup.

## 2.2. Biểu đồ Usecase

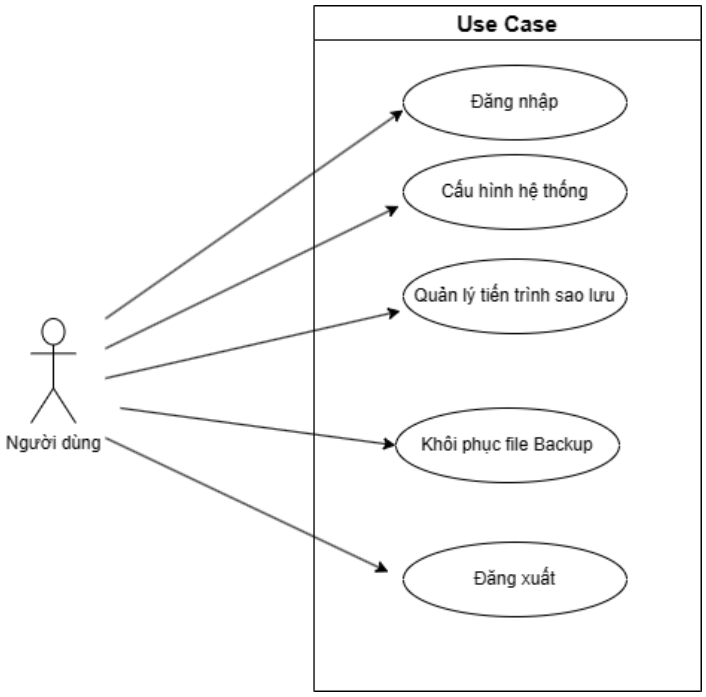
### 2.2.1 Danh sách các usecase hệ thống

*Bảng 2.1 Danh sách các usecase hệ thống*

| Use Case | Đối tượng |
|---|---|
| Đăng nhập | Người dùng |
| Cấu hình hệ thống | Người dùng |
| Tạo mới sao lưu | Người dùng |

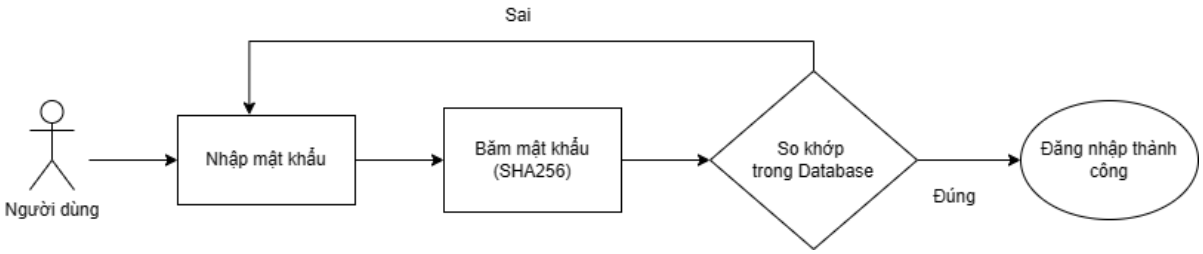
| | |
|---|---|
| Cập nhật sao lưu | Người dùng |
| Xem sao lưu | Người dùng |
| Xóa sao lưu | Người dùng |
| Xuất dữ liệu | Người dùng |
| Chạy sao lưu | Người dùng |
| Đăng xuất | Người dùng |

### 2.2.2 Biểu đồ usecase tổng quát




*Hình 2.2 Biểu đồ usecase tổng quát*

### 2.2.3 Chức năng đăng nhập



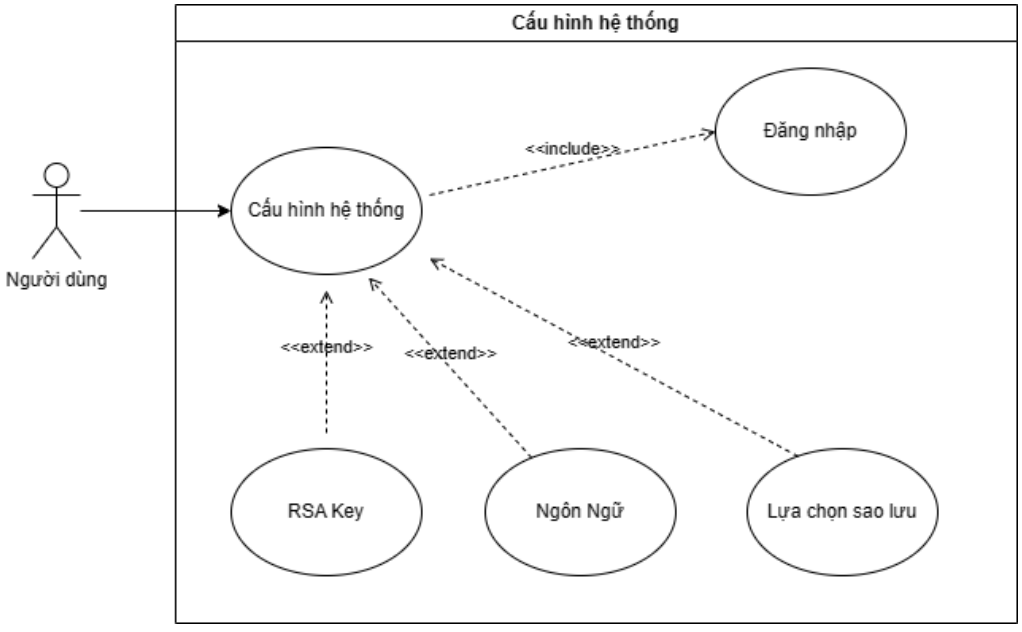

*Hình 2.3 Biểu đồ usecase đăng nhập*

Đặc tả usecase đăng nhập:

*Bảng 2.2 Đặc tả usecase đăng nhập*

| **Đối tượng** | User |
|---|---|
| **Điều kiện** | Người dùng đăng nhập vào ứng dụng bằng password<br>Người dùng chưa đăng nhập và ứng dụng bật tính năng yêu cầu mật khẩu |
| **Luồng hoạt động** | 1. Người dùng vào giao diện ứng dụng<br>2. Người dùng nhập "Password"<br>3. Người dùng nhấn nút "Sign in"<br>4. Hệ thống tả về kết quả đăng nhập và chuyển đến màn hình Home |
| **Thông báo lỗi** | "Login failed: Unauthorized" |

### 2.2.4 Chức năng cấu hình hệ thống



*Hình 2.4 Biểu đồ usecase cấu hình hệ thống*

a) Đặc tả usecase sinh khóa RSA

*Bảng 2.3 Đặc tả usecase sinh khóa RSA*

| **Đối tượng** | User |
|---|---|
| **Mô tả** | Người dùng tiến hành thiết lập hệ thống trước khi sao lưu và khôi phục |
| **Điều kiện** | Người dùng đăng nhập |
| **Luồng hoạt động** | 1. Người dùng vào giao diện trang cài đặt<br>2. Người dùng bấm nút "Generate" để sinh cặp khóa RSA<br>3. Người dùng lựa chọn độ lớn của khóa<br>4. Người dùng bấm nút "OK" để lưu cài đặt<br>5. Hệ thống tiến hành lưu lại thông tin |
| **Thông báo lỗi** | "Không được để trống trường thông tin này" |

b) Đặc tả usecase cài đặt ngôn ngữ

*Bảng 2.4 Đặc tả usecase cài đặt ngôn ngữ*

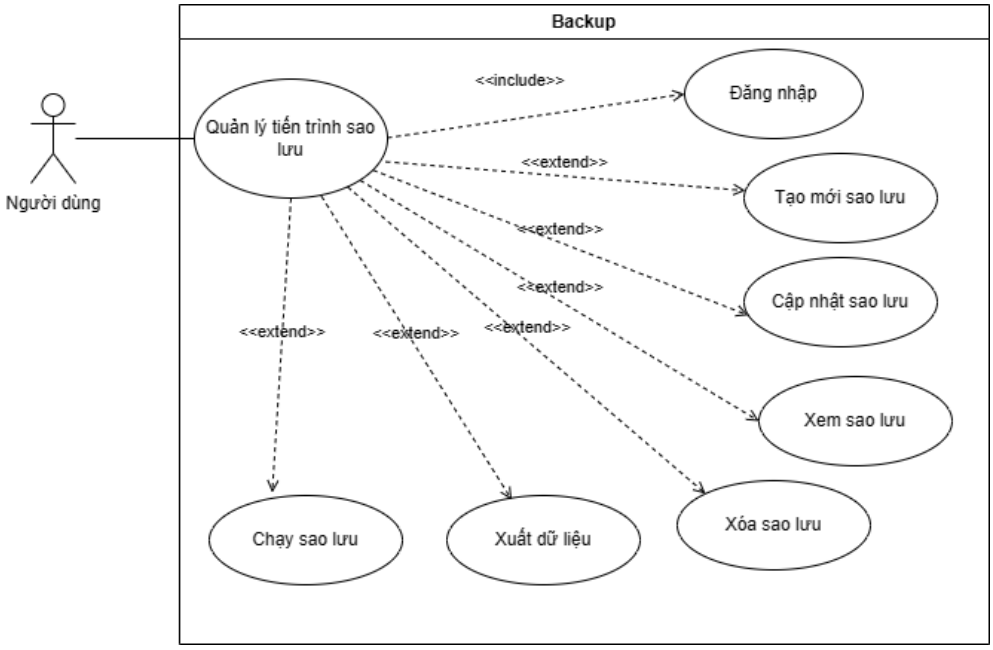
| **Đối tượng** | User |
|---|---|
| **Mô tả** | Người dùng tiến hành thiết lập hệ thống trước khi sao lưu và khôi phục |
| **Điều kiện** | Người dùng đăng nhập |
| **Luồng hoạt động** | 1. Người dùng vào giao diện trang cài đặt<br>2. Người dùng đến mục "User interface settings" xổ xuống và chọn ngôn ngữ<br>3. Người dùng bấm nút "OK" để lưu cài đặt<br>4. Hệ thống tiến hành lưu lại thông tin |
| **Thông báo lỗi** | |

c) Đặc tả usecase lựa chọn sao lưu

*Bảng 2.5 Đặc tả usecase lựa chọn sao lưu*

| **Đối tượng** | User |
|---|---|
| **Mô tả** | Người dùng tiến hành thiết lập hệ thống trước khi sao lưu và khôi phục |
| **Điều kiện** | Người dùng đăng nhập |
| **Luồng hoạt động** | 1. Người dùng vào giao diện trang cài đặt<br>2. Người dùng đến mục "Default Option" xổ xuống và pick an option<br>3. Tiến hành các cài đặt muốn cài thêm<br>4. Người dùng bấm nút "OK" để lưu cài đặt<br>5. Hệ thống tiến hành lưu lại thông tin |
| **Thông báo lỗi** | |

### 2.2.5 Chức năng quản lý tiến trình sao lưu




*Hình 2.5 Biểu đồ usecase quản lý tiến trình sao lưu*

a) Đặc tả usecase tạo mới sao lưu

*Bảng 2.6 Đặc tả usecase tạo mới sao lưu*

| **Đối tượng** | User |
|---|---|
| **Mô tả** | Người dùng tạo mới bản sao lưu |
| **Điều kiện** | Người dùng đã đăng nhập, đã cài đặt cặp khóa RSA |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng chọn "Add backup"<br>2. Hệ thống hiển thị form để người dùng chọn thêm cấu hình mới hoàn toàn hoặc import từ file JSON<br>3. Người dùng chọn phương thức thêm mới<br>4. Hệ thống hiển thị các cài đặt về tiến trình backup để người dùng nhập thông tin<br>5. Người dùng nhấn nút "Save"<br>6. Hệ thống tạo mới tiến trình và quay về màn hình Home |
| **Thông báo lỗi** | |

b) Đặc tả usecase cập nhật sao lưu

*Bảng 2.7 Đặc tả usecase cập nhật sao lưu*

| **Đối tượng** | User |
|---|---|
| **Mô tả** | Người dùng tạo cập nhật sao lưu |
| **Điều kiện** | Người dùng đã đăng nhập |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng chọn mục "Home"<br>2. Hệ thống hiển thị thông tin các tiến trình sao lưu hiện có |

| | |
|---|---|
| | 3. Người dùng chọn tiến trình cần cập nhật và nhấn nút "Edit"<br>4. Hệ thống hiển thị các cài đặt về tiến trình backup để người dùng cập nhật thông tin<br>5. Người dùng nhấn nút "Save"<br>6. Hệ thống cập nhật tiến trình và quay về màn hình Home |
| **Thông báo lỗi** | |

c) Đặc tả usecase xem sao lưu

*Bảng 2.8 Đặc tả usecase xem sao lưu*

| | |
|---|---|
| **Đối tượng** | User |
| **Mô tả** | Người dùng xem các tiến trình sao lưu đã tạo. |
| **Điều kiện** | Người dùng đăng nhập, đã cài đặt cặp khóa RSA |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng chọn mục "Home"<br>2. Hệ thống hiển thị thông tin các tiến trình sao lưu hiện có |
| **Thông báo lỗi** | |

d) Đặc tả usecase xóa sao lưu

*Bảng 2.9 Đặc tả usecase xóa sao lưu*

| | |
|---|---|
| **Đối tượng** | User |
| **Mô tả** | Người dùng xoá tiến trình sao lưu đã tạo. |
| **Điều kiện** | Người dùng đã đăng nhập |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng chọn mục "Home"<br>2. Hệ thống hiển thị thông tin các tiến trình sao lưu hiện có |

| | |
|---|---|
| | 3. Người dùng chọn tiến trình cần cập nhật và nhấn nút "Delete"<br>4. Hệ thống hiển thị màn hình xác nhận xoá tiến trình<br>5. Người dùng nhấn nút "Delete backup"<br>6. Hệ thống xoá tiến trình và quay về màn hình Home |
| **Thông báo lỗi** | |

e) Đặc tả usecase xuất dữ liệu sao lưu

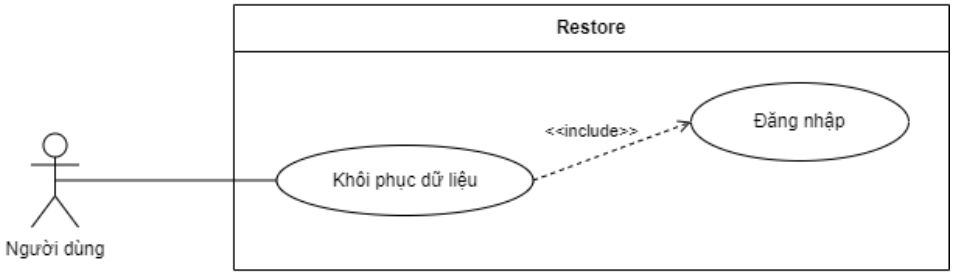
*Bảng 2.10 Đặc tả usecase xuất dữ liệu sao lưu*

| **Đối tượng** | User |
|---|---|
| **Mô tả** | Người dùng xoá tiến trình sao lưu đã tạo. |
| **Điều kiện** | Người dùng đăng nhập, đã cài đặt cặp khóa RSA |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng chọn mục "Home"<br>2. Hệ thống hiển thị thông tin các tiến trình sao lưu hiện có<br>3. Người dùng chọn tiến trình cần export dữ liệu và nhấn nút "Export"<br>4. Hệ thống hiển thị màn hình lựa chọn cách thức export<br>5. Người dùng chọn cách thức export và nhấn nút "Export"<br>6. Hệ thống export dữ liệu và hiển thị kết quả |
| **Thông báo lỗi** | |

f) Đặc tả usecase chạy sao lưu

| **Đối tượng** | User |
|---|---|
| **Mô tả** | Người dùng chạy tiến trình sao lưu dữ liệu thủ công hoặc theo lịch trình. |
| **Điều kiện** | Người dùng đã đăng nhập |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng chọn mục "Home"<br>2. Hệ thống hiển thị thông tin các tiến trình sao lưu hiện có<br>3. Người dùng chọn tiến trình cần export dữ liệu và nhấn nút "Run now"<br>4. Hệ thống chạy tiến trình sao lưu.<br>5. Trong trường hợp tiến trình đã được lập lịch, hệ thống sẽ chạy tiến trình theo lịch đã tạo |
| **Thông báo lỗi** | |

### 2.2.6 Chức năng khôi phục dữ liệu




*Hình 2.6 Biểu đồ usecase khôi phục dữ liệu*

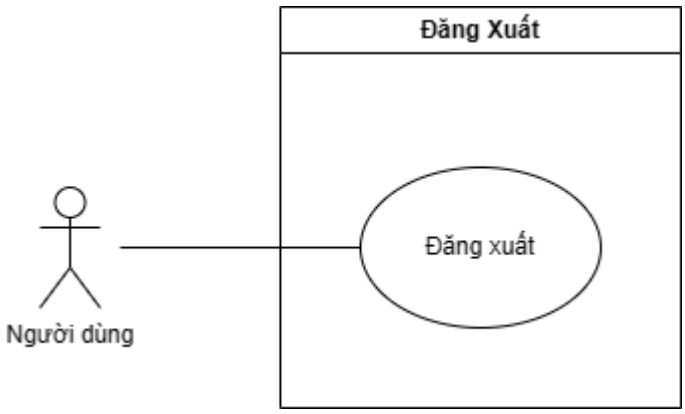
Đặc tả usecase khôi phục dữ liệu:

*Bảng 2.11 Đặc tả usecase khôi phục dữ liệu*

| **Đối tượng** | User |
|---|---|

| | |
|---|---|
| **Mô tả** | Người dùng khôi phục dữ liệu từ bản sao lưu. |
| **Điều kiện** | Người dùng đã đăng nhập |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng chọn mục "Restore"<br>2. Hệ thống hiển thị các cách thức khôi phục dữ liệu<br>3. Người dùng chọn option và nhấn nút "Continue"<br>4. Người dùng tiến hành chọn vị trí lưu bản dữ liệu<br>5. Hệ thống khôi phục dữ liệu cho người dùng |
| **Thông báo lỗi** | |

### 2.2.7 Chức năng đăng xuất




*Hình 2.7 Biểu đồ usecase đăng xuất*

Đặc tả usecase đăng xuất:

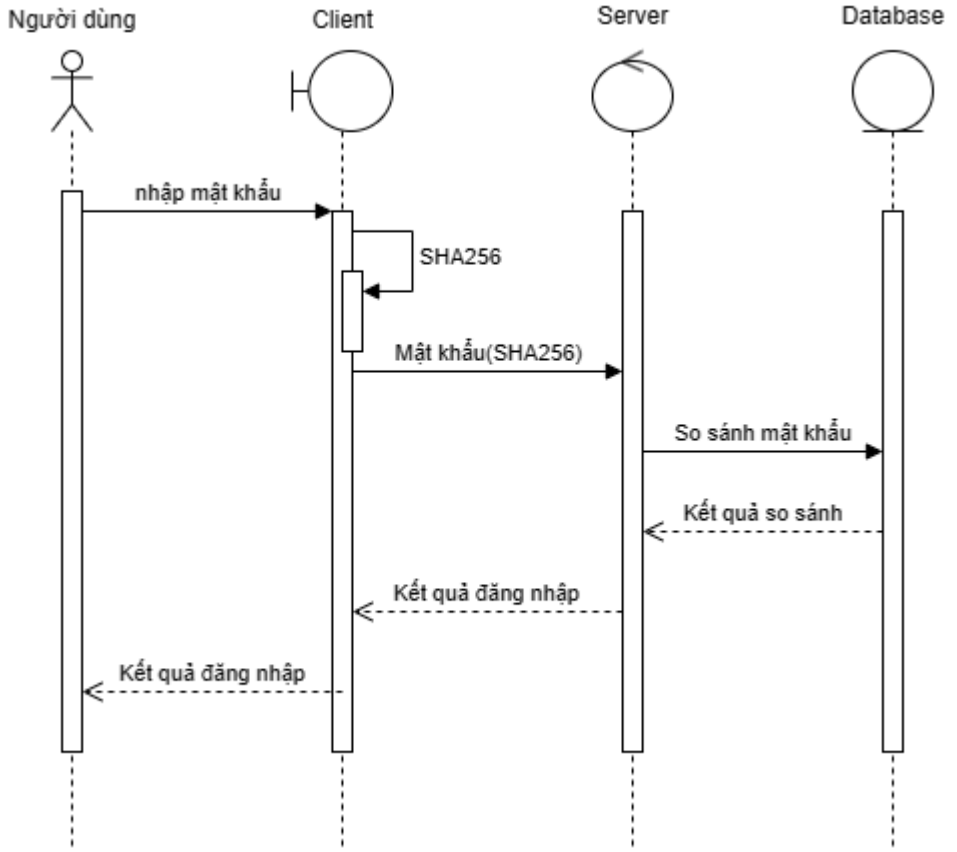
*Bảng 2.12 Đặc tả usecase đăng xuất*

| | |
|---|---|
| **Đối tượng** | User |
| **Mô tả** | Người dùng đăng xuất khỏi ứng dụng. |
| **Điều kiện** | Người dùng đã đăng nhập |
| **Luồng hoạt động** | 1. Trong phần sidebar, người dùng nhấn nút "Log out" |

| | |
|---|---|
| | 2. Hệ thống quay về màn hình đăng nhập |
| **Thông báo lỗi** | |

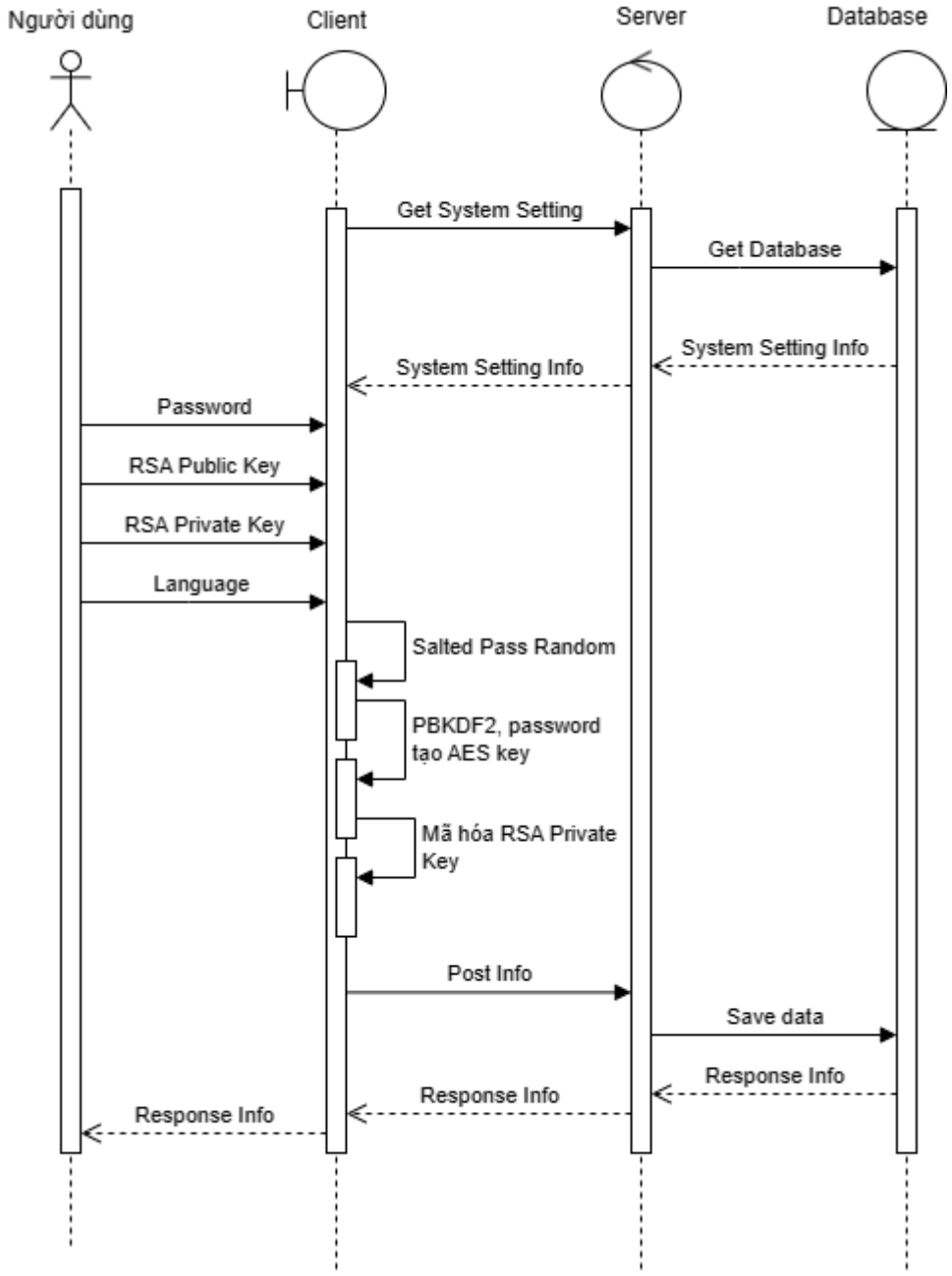
## 2.3. Biểu đồ tuần tự

### 2.3.1 Chức năng đăng nhập




*Hình 2.8 Biểu đồ tuần tự chức năng đăng nhập*

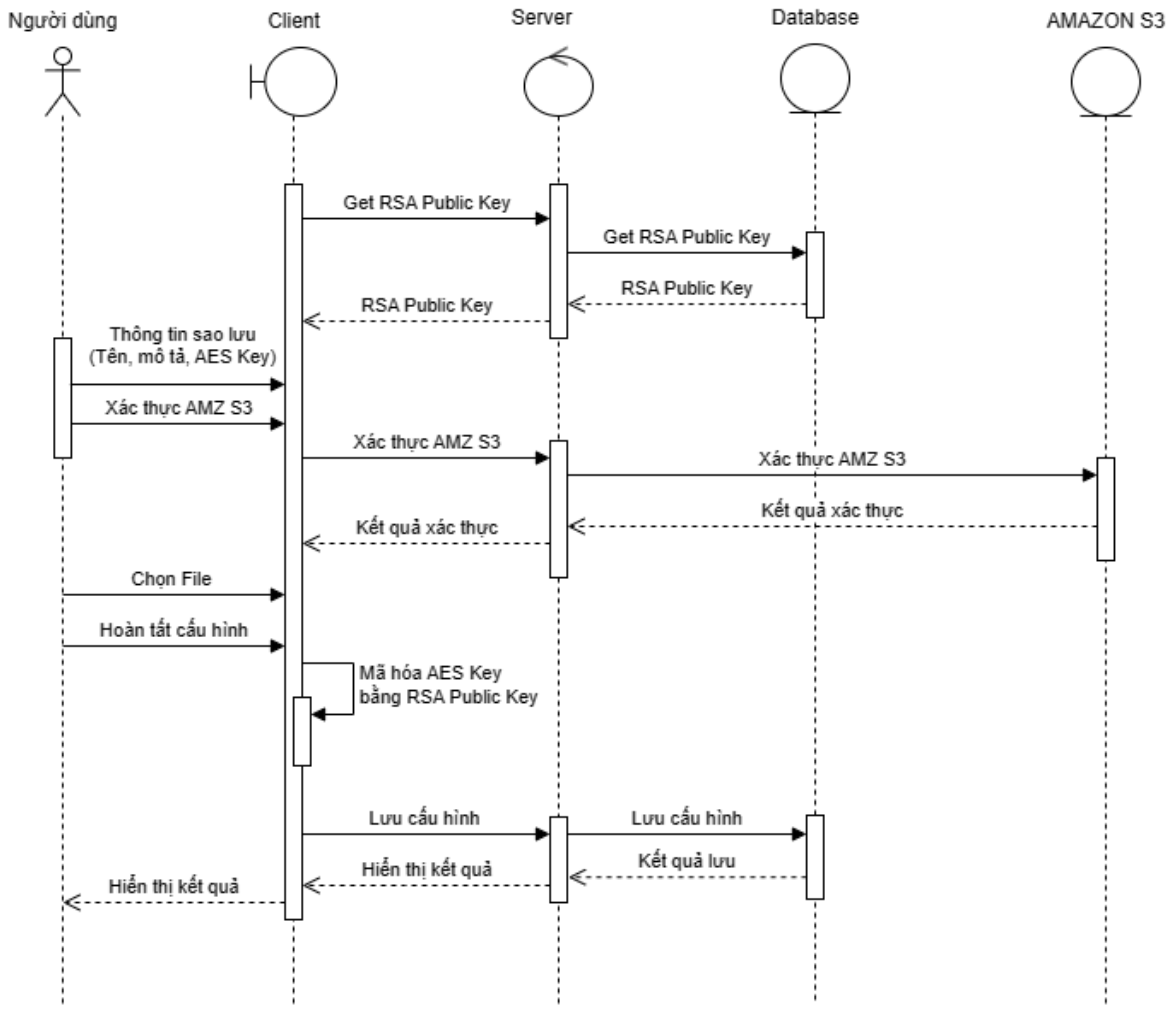
### 2.3.2 Chức năng cấu hình hệ thống




*Hình 2.9 Biểu đồ tuần tự chức năng cấu hình hệ thống*

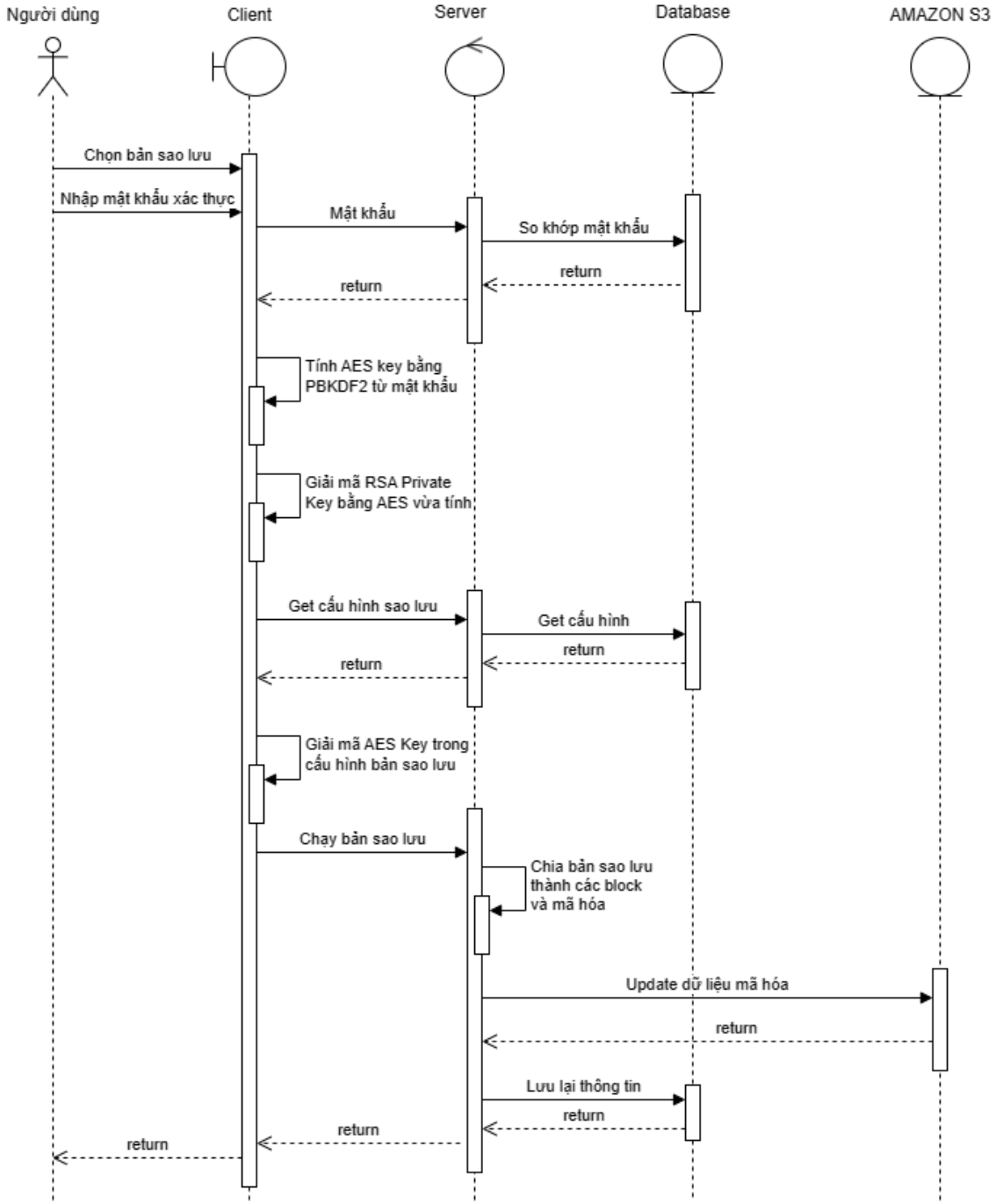
### 2.3.3 Chức năng quản lý tiến trình sao lưu

a) Chức năng tạo mới sao lưu



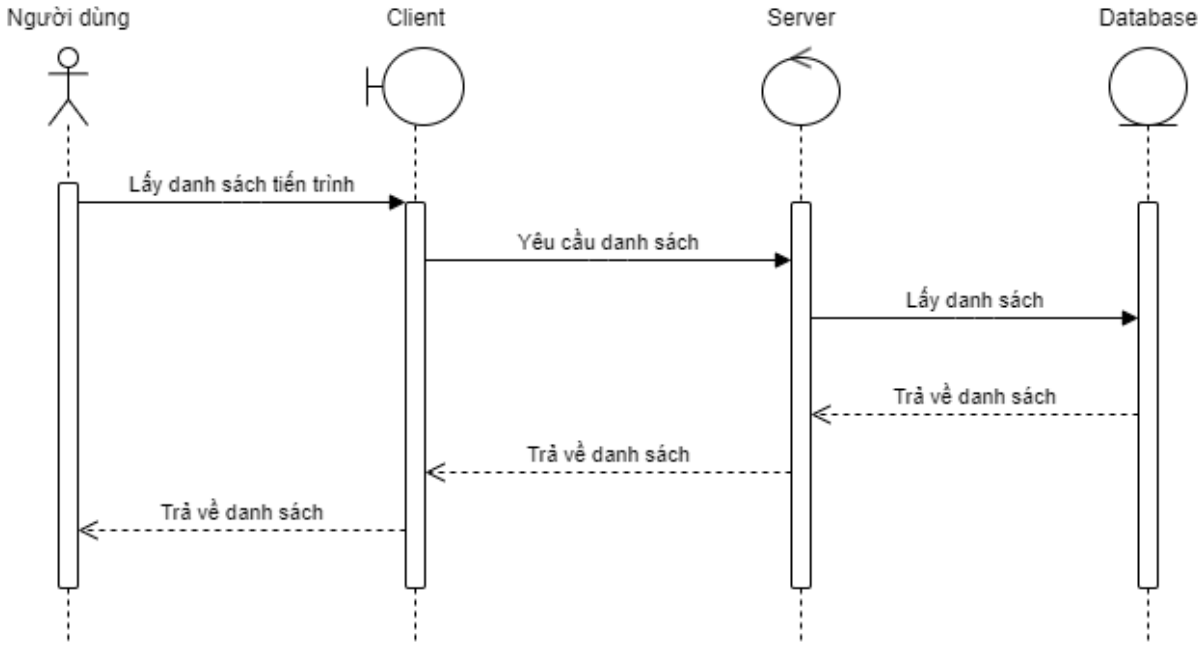

*Hình 2.10 Biểu đồ tuần tự chức năng tạo mới sao lưu*

b) Chức năng chạy sao lưu



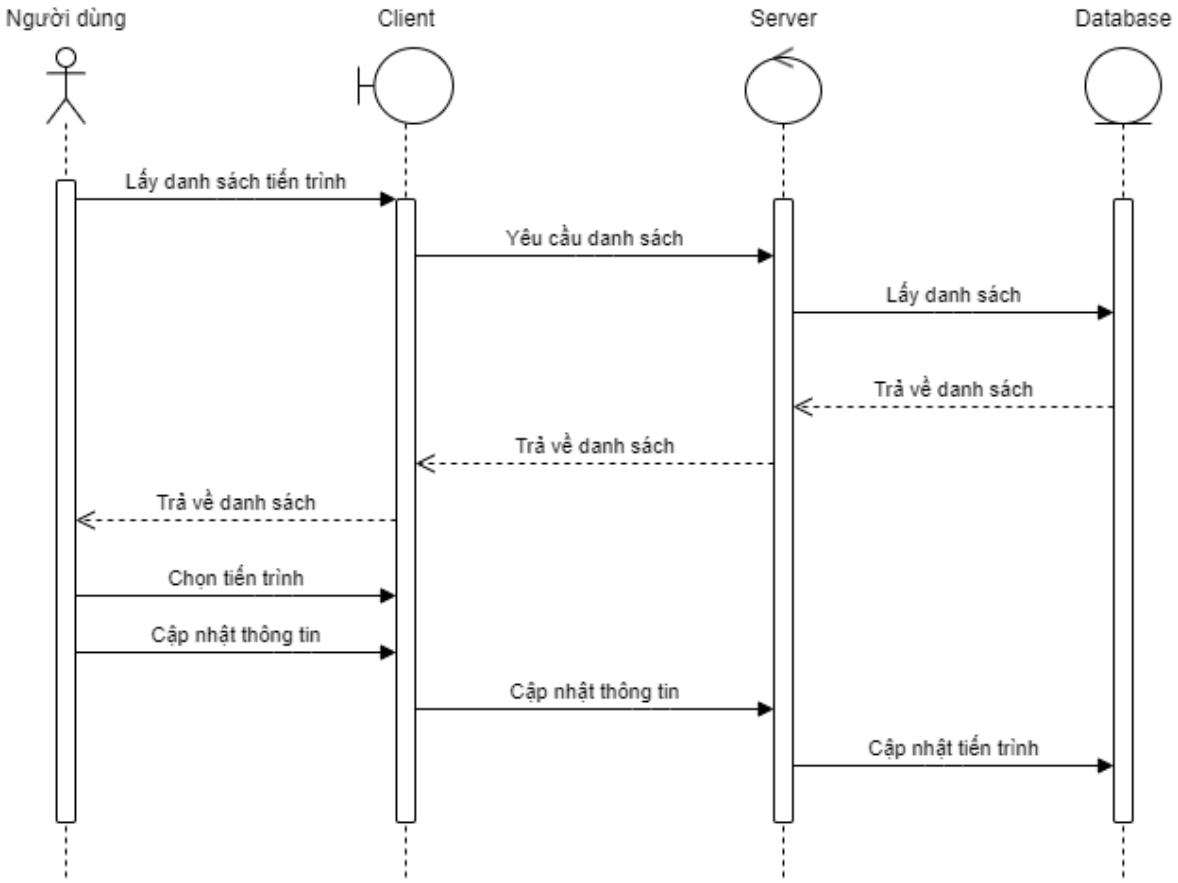

*Hình 2.11 Biểu đồ tuần tự chức năng chạy sao lưu*

c) Chức năng xem sao lưu



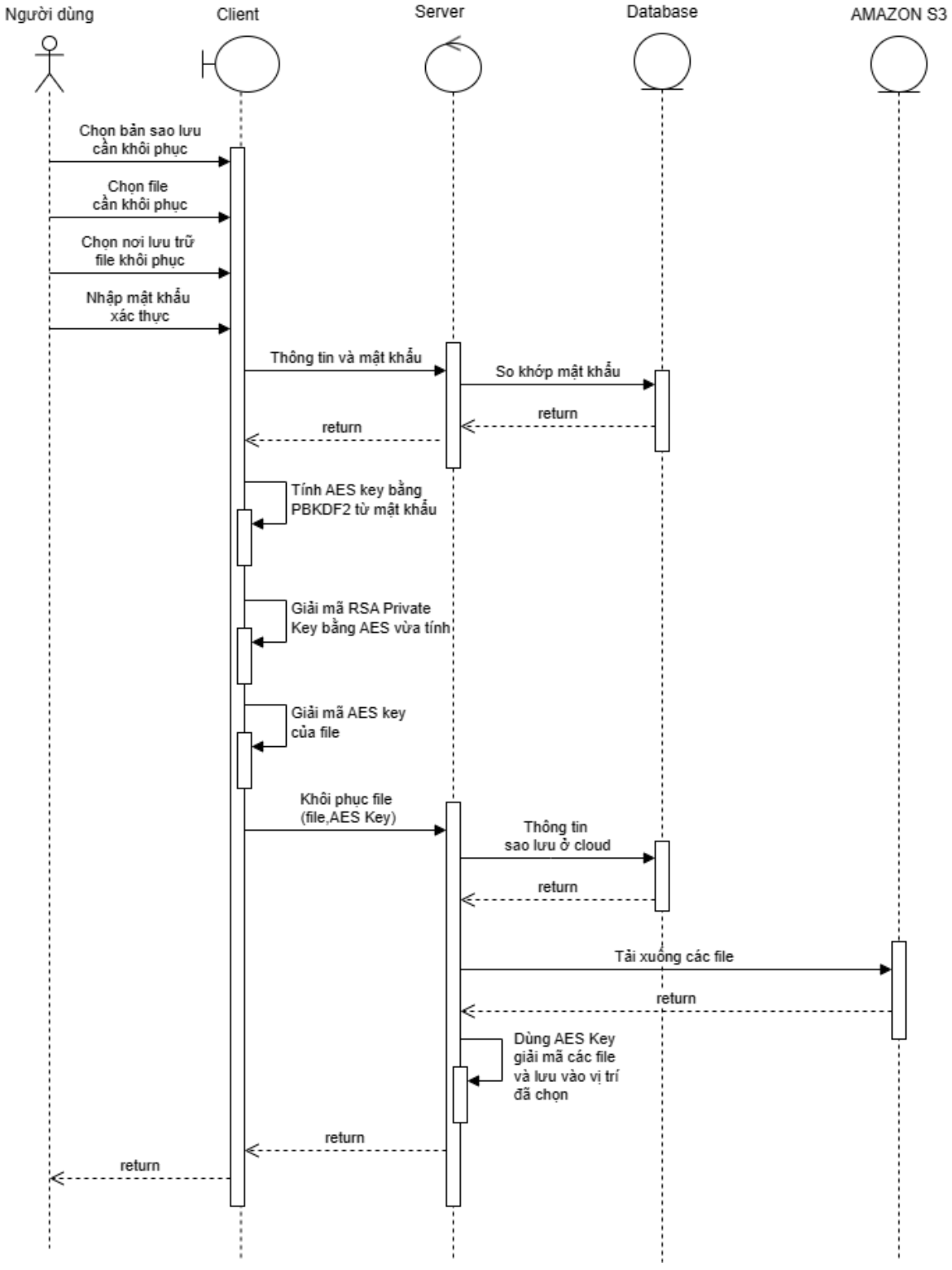

*Hình 2.12 Biểu đồ tuần tự chức năng xem sao lưu*

d) Chức năng cập nhật tiến trình



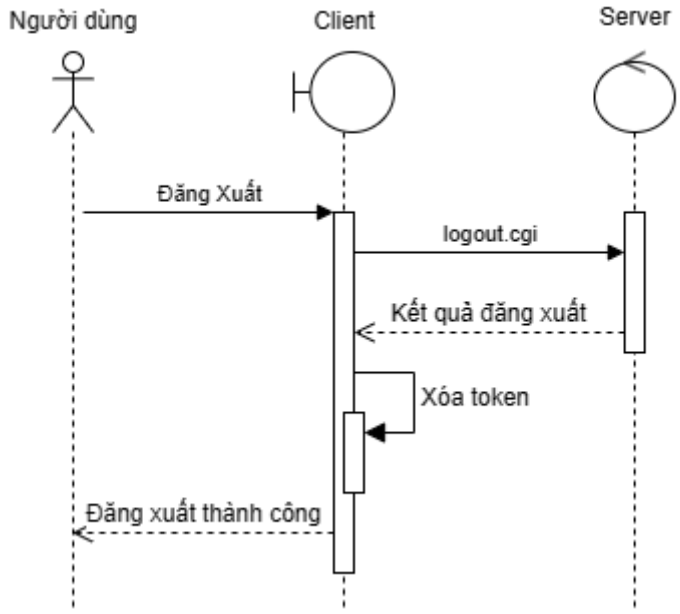

*Hình 2.13 Biểu đồ tuần tự chức năng cập nhật tiến trình*

### 2.3.4 Chức năng khôi phục dữ liệu



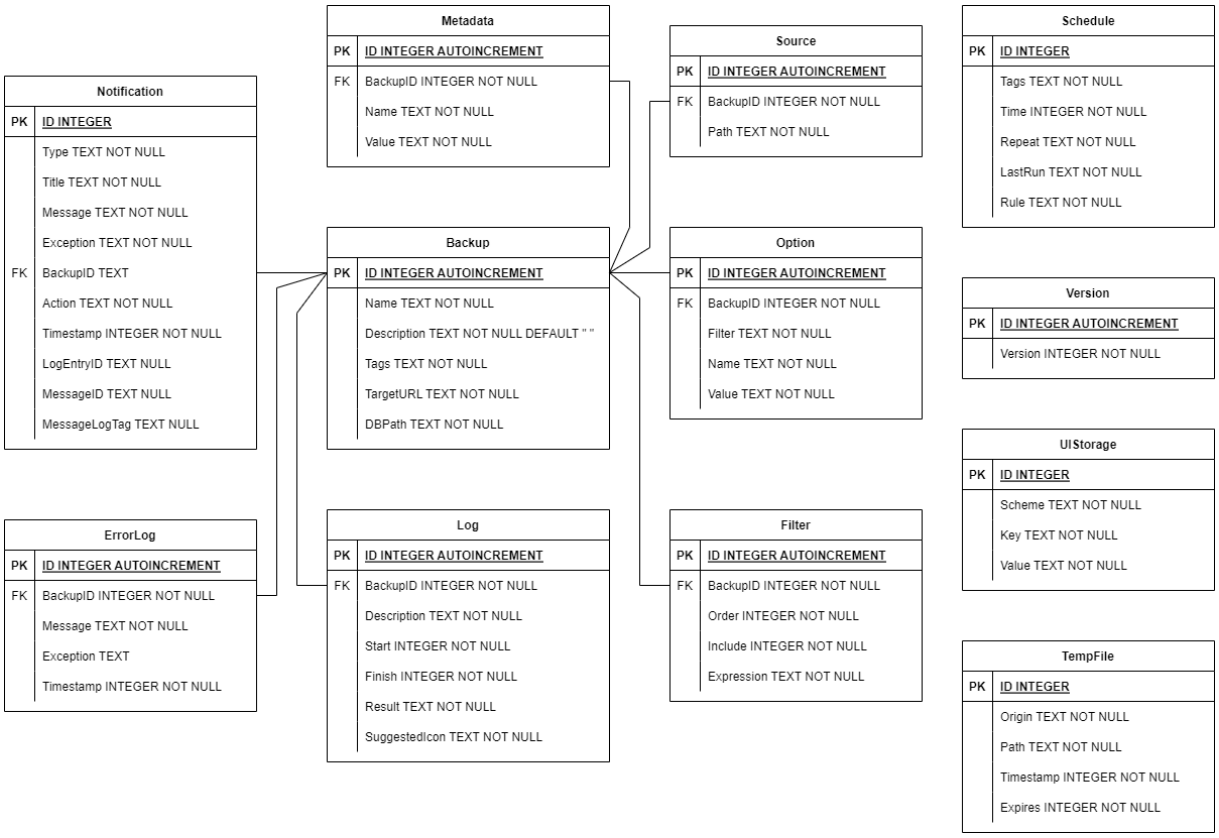

*Hình 2.14 Biểu đồ tuần tự chức năng khôi phục dữ liệu*

### 2.3.5 Chức năng đăng xuất




*Hình 2.15 Biểu đồ tuần tự chức năng đăng xuất*

## 2.4. Thiết kế cơ sở dữ liệu




*Hình 2.16 Sơ đồ thiết kế cơ sở dữ liệu của Duplicati*

Duplicati sử dụng hệ quản trị cơ sở dữ liệu SQLite, bao gồm các bảng:

- Backup: bảng chính lưu thông tin sao lưu dữ liệu, gồm các trường:
- Schedule: lưu các tác vụ thực hiện theo lịch trình đặt sẵn
- Source: lưu danh sách thư mục và file nguồn để sao lưu
- Filter: lưu thông tin bộ lọc dành cho quá trình sao lưu, nếu trường backupID bằng $-1$ thì nó áp dụng cho toàn bộ các sao lưu
- Option: các tuỳ chọn khi sao lưu
- Metadata: lưu thông tin đính kèm của sao lưu, chẳng hạn như lần cuối thực hiện sao lưu, thời gian thực thi, số file, tổng dung lượng file…
- Notification: lưu các thông báo tới người dùng
- Log: lưu các tác vụ thực hiện bởi module lập lịch hay người dùng
- ErrorLog: lưu thông tin lỗi xảy ra trong quá trình sao lưu
- Version: thông tin phiên bản của Duplicati
- UIStorage: lưu dữ liệu dạng key – value cho frontend
- TempFile: lưu các file tạm cần lưu trữ trong thời gian dài

## 2.5. Tổng kết chương

Trong chương này, đồ án đã trình bày một cái nhìn tổng quan về kiến trúc của hệ thống sao lưu và phục hồi dữ liệu. Các thành phần chính bao gồm giao diện người dùng, giao diện dòng lệnh và máy chủ tương tác với các dịch vụ lưu trữ. Qua đó, đồ án đã chi tiết hóa kiến trúc này thông qua việc phân tích và mô tả các biểu đồ Use Case và Tuần tự.

Biểu đồ Use Case đã giúp hiểu rõ các chức năng chính của hệ thống, bao gồm sao lưu dữ liệu, phục hồi dữ liệu và quản lý cấu hình hệ thống. Điều này làm nổi bật những hoạt động mà người dùng có thể thực hiện và cách họ tương tác với hệ thống.Sau đó, để minh họa quá trình xử lý và tương tác giữa các thành phần, đồ án đã xây dựng các biểu đồ tuần tự. Điều này giúp có cái nhìn rõ ràng về luồng làm việc từng bước một của hệ thống.

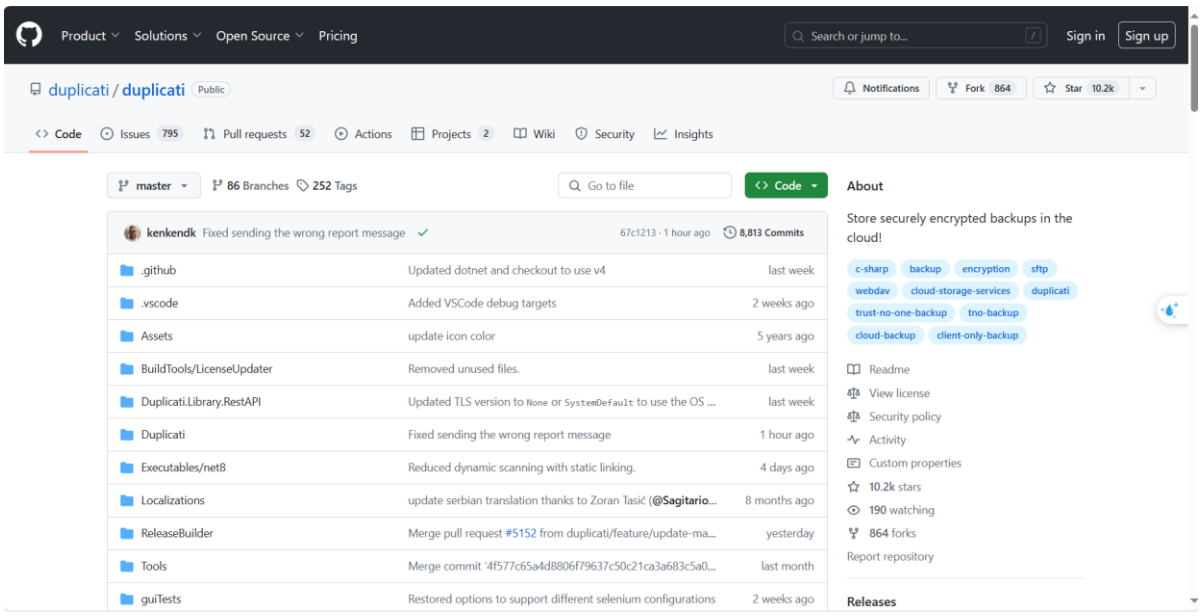
Cuối cùng, đồ án đã thiết kế cơ sở dữ liệu với các bảng và quan hệ giữa chúng để lưu trữ thông tin liên quan đến quá trình sao lưu và phục hồi dữ liệu.

# CHƯƠNG 3. XÂY DỰNG VÀ THỬ NGHIỆM HỆ THỐNG

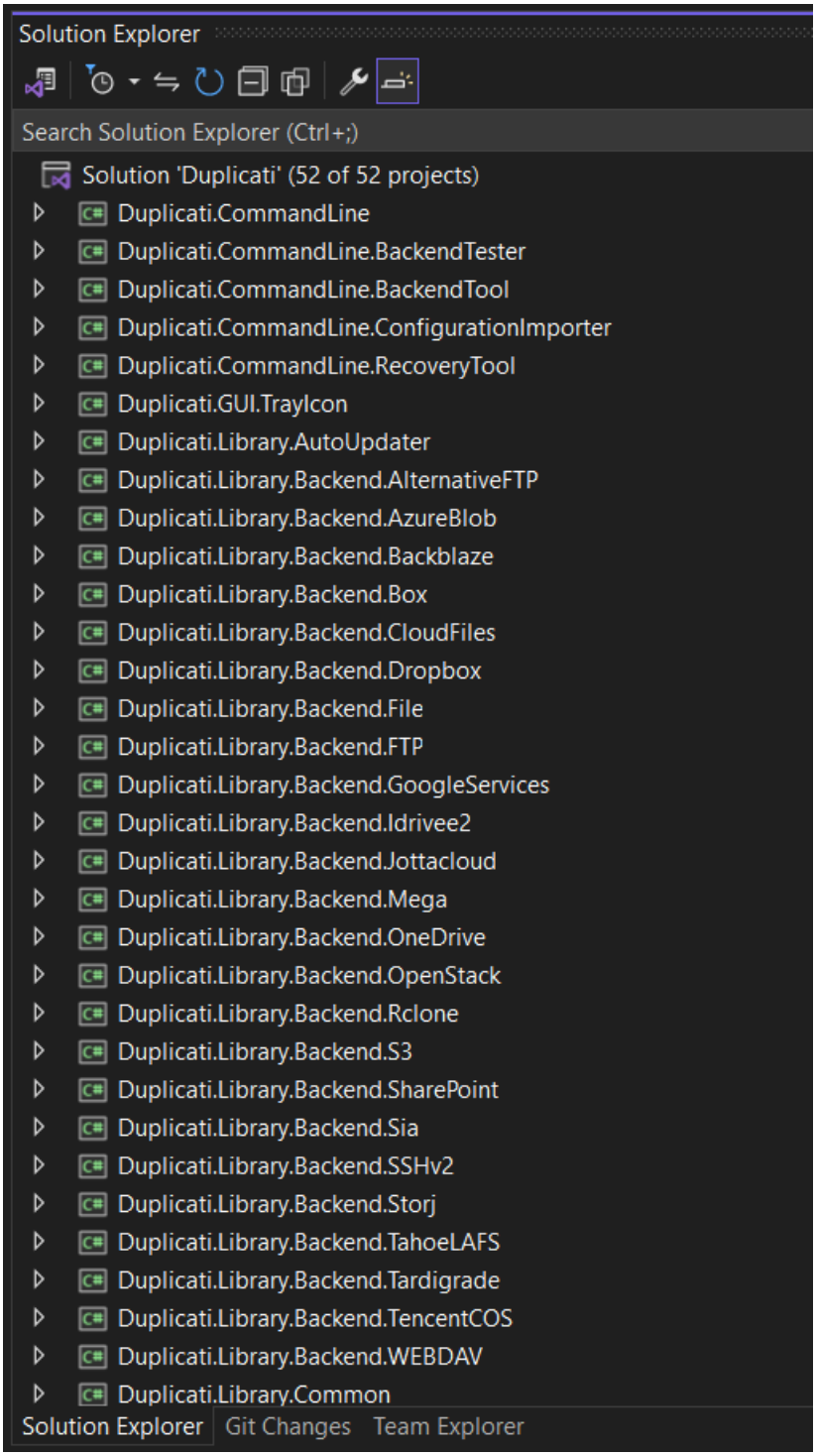
## 3.1. Phân tích mã nguồn Duplicati

### 3.1.1 Cấu trúc mã nguồn Duplicati

- Duplicati là giải pháp sao lưu mã nguồn mở miễn phí. Duplicati hỗ trợ trên hầu hết các hệ điều hành bao gồm: Windows, Linux, Mac.
- Duplicati hỗ trợ phương pháp mã hóa AES-256 và có thể kết nối với nhiều công cụ lưu trữ của bên thứ 3 như: Google Drive, Microsoft OneDrive, Amazon Cloud Drive.
- Duplicati hỗ trợ sử dụng qua cả giao diện web và giao diện dòng lệnh CLI. Người dùng có thể thiết lập các cấu hình backup định kỳ theo thời gian và khôi phục file một cách thủ công qua giao diện thân thiện.
- Để có thể phát triển được giải pháp đã thiết kế và mô tả ở trên, cần phải nắm rõ chi tiết các thành phần bên trong của hệ thống Duplicati để có thể tùy chỉnh từ mã nguồn theo ý muốn:
- Project Duplicati trên github tại: https://github.com/duplicati/duplicati




*Hình 3.1 Mã nguồn dự án Duplicati*




*Hình 3.2 Cấu trúc mã nguồn Duplicati*

Hệ thống của Duplicati chia ra các thành phần chính bao gồm:

- Library: chứa tất cả các thư viện, class và module được sử dụng.

- Backend: thư mục chứa mã nguồn xử lý các phương pháp sao lưu và khôi phục dữ liệu từ các nền tảng lưu trữ khác nhau như Amazon S3, Microsoft OneDrive, Google Drive, FTP…
- Common: thực hiện các thao tác chung với file như lấy danh sách file, tên file, đường dẫn, kích thước, ngày sửa đổi cuối, quyền truy cập, loại (file hay thư mục).
- Compression: xử lý các thao tác nén dữ liệu.
- Encryption: chứa các file xử lý thao tác mã hoá theo chuẩn AES-256.
- Interface: định nghĩa các interface dùng trong ứng dụng.
- Main: chứa các module chính của Duplicati, bao gồm các phương thức để quản lý công việc sao lưu và phục hồi dữ liệu.
- Snapshots: xử lý việc tạo snapshot để quản lý các phiên bản sao lưu.
- SQLiteHelper: các hàm hỗ trợ tương tác với cơ sở dữ liệu như đặt biến môi trường, kết nối cơ sở dữ liệu, kiểm tra kết nối hay ngắt kết nối.
- Utility: các thao tác hỗ trợ như biên dịch câu lệnh trên CLI, tính toán hàm băm, worker thread để chạy đa luồng…
- Server: thư mục chứa mã nguồn liên quan đến máy chủ Duplicati, bao gồm các thành phần như giao diện web, xử lý request từ người dùng và quản lý cấu hình
- Webroot: gồm các file HTML, CSS và JavaScript để tạo giao diện web
- WebServer: khởi tạo máy chủ Duplicati, định nghĩa các REST API để tương tác với giao diện web.
- Database: chứa các file xử lý và tương tác với cơ sở dữ liệu.
- Duplicati-CommandLine: các chức năng thông qua giao diện dòng lệnh Duplicati-GUI: Project khởi tạo chạy các thành phần trên Windows. Nếu sử dụng môi trường Windows thì project này sẽ chạy và khởi tạo tất cả các thành phần khác để hệ thống hoạt động được.

### 3.1.2 Module S3

Module S3 trong Duplicati cho phép người dùng thực hiện sao lưu dữ liệu của mình lên một bucket của dịch vụ Amazon S3. Một điểm cần lưu ý khi sử dụng Amazon S3 đó là khi người dùng nhập vào các giá trị BucketName, AccessKeyId và SecretAccessKey, cần đảm bảo các key đó được gắn với một tài khoản có đầy đủ các quyền PutObject và GetObject đối với S3 bucket, bao gồm cả identity-based policy và resource-based: policy. Nếu không, quá trình sao lưu và phục hồi sẽ gặp lỗi từ chối quyền truy cập và không thực hiện được.

Trong mã nguồn Duplicati, module tương tác với S3 được đặt trong thư mục *Duplicati\Library\Backend\S3*, bao gồm các file chính sau:

***S3Backend.cs***: đây là file chính liên quan đến xử lý S3 backend trong Duplicati. Nó cung cấp các phương thức và thuộc tính để tương tác với dịch vụ lưu trữ S3.

Các thuộc tính:

***KNOWN_S3_PROVIDERS***: các bên cung cấp dịch vụ S3.

***KNOWN_S3_LOCATIONS***: các region của S3.

**DEFAULT_S3_LOCATION_BASED_HOSTS**: URL của region.

***KNOWN_S3_STORAGE_CLASSES***: các storage class của S3. Người dùng cần lựa chọn storage class phù hợp để tiết kiệm chi phí lưu trữ.

Các phương thức: tạo bucket, tạo file, đọc, ghi và xóa file trên S3

ListBucket: lấy danh sách các bucket có trên S3.

AddBucket: tạo mới bucket.

DeleteObject: xoá file trong bucket.

RenameFile: đổi tên file.

GetFileStream: tải xuống file.

GetDnsHost: lấy URL của S3 bucket.

AddFileStreamAsync: tải file lên S3.

***S3AwsClient.cs***, ***S3MinioClient.cs***: các file chứa các hàm tương tác với S3, được gọi đến từ *S3Backend.cs*.

***S3Config.cs***: lấy thông tin cấu hình của S3 về các nhà cung cấp, region, URL tương ứng với region, storage class.

***S3IAM.cs***: tạo IAM user để có quyền tương tác với S3. Để có thể tạo được user, ta cần AccessKeyId và SecretAccessKey của một tài khoản có quyền tạo IAM user.

***IS3Client.cs***: interface của S3 client.

## 3.2. Xây dự hệ thống sao lưu và khôi phục

Xây dựng bổ sung tính năng tại các vị trí:

- Giao diện
- Cấu hình hệ thống
- Chạy cấu hình sao lưu
- Chạy khôi phục file

### 3.2.1 Chỉnh sửa giao diện

File giao diện: Duplicati\Server\webroot\ngax\templates\settings.html

File Backend: Duplicati\Server\WebServer\RESTMethods\ServerInfo.cs

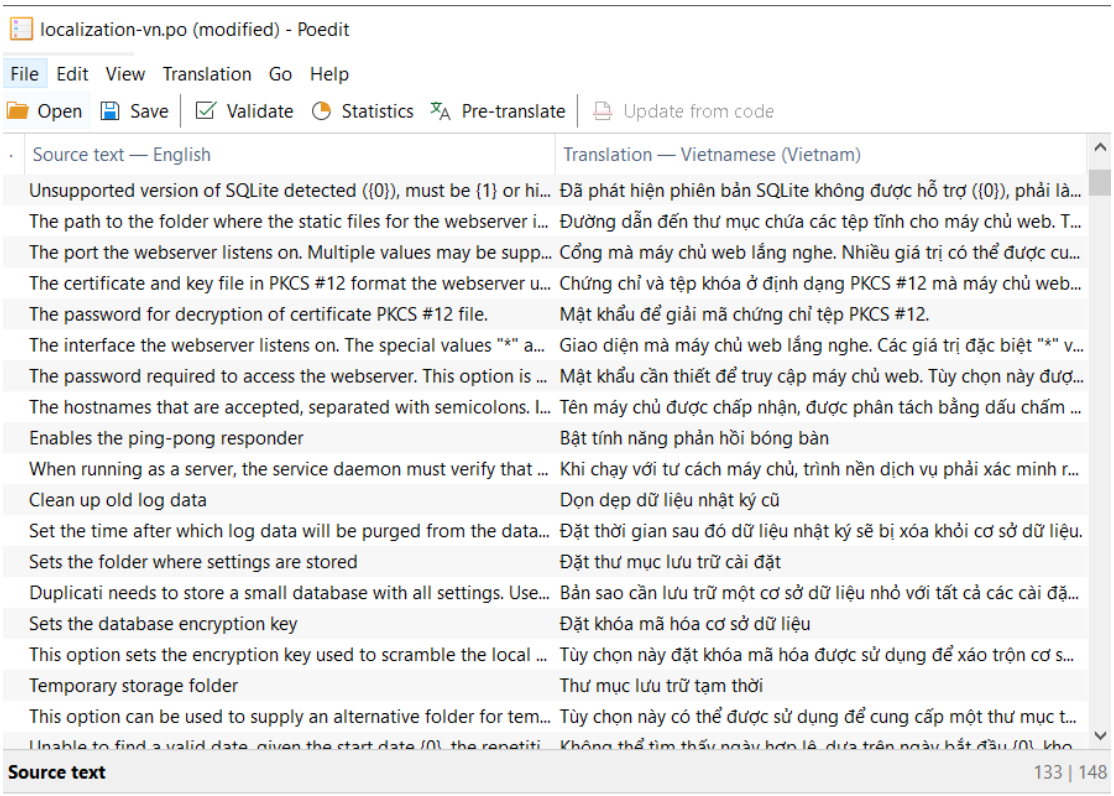
Duplicati\Library\Localization\MoLocalizationService.cs

Ở phía Backend sẽ đọc các file .PO và .MO theo các ngôn ngữ đã cài đặt sẵn trong file và hiển thị lên giao diện.

```
private static IEnumerable<CultureInfo> SupportedCultureInfos
{
     get {
         if (m_supportedcultures == null) {
             var lst = new string[0].AsEnumerable();
             if (SearchAssembly != null)
                 lst =
lst.Union(SearchAssembly.GetManifestResourceNames());
             foreach (var sp in SearchPaths)
                 if (Directory.Exists(sp))
                     lst = lst.Union(Directory.GetFiles(sp,
"localization-*.mo"));
             var allcultures =
                 from name in lst
                 let m = CI_MATCHER.Match(name)
                 let ci = m.Success ?
LocalizationService.ParseCulture(m.Groups["culture"].Value) :
null
                 where ci != null
```

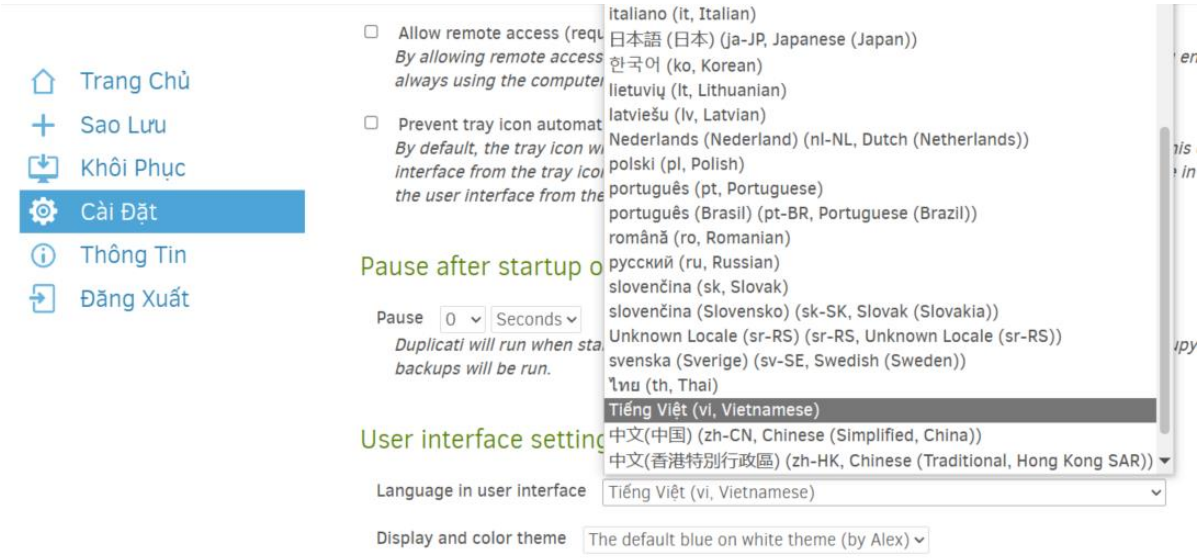
```
                    select ci;
                m_supportedcultures = allcultures.Concat(new[] {
new CultureInfo("en") })
                .Distinct();
            }
            return m_supportedcultures;
        }
    }
```

Dùng Poedit để tạo ra các file .po và .mo và thêm vào đường dẫn Localization\duplicati và Localization\webroot




*Hình 3.3 Poedit tạo ra các file .mo .po*

Sau khi thiết lập như trên thì ở phía giao diện người dùng ở phần settings sẽ có thêm lựa chọn ngôn ngữ tiếng việt và chọn thì giao diện sẽ trở thành tiếng việt.




*Hình 3.4 Thiết lập cấu hình cài đặt ngôn ngữ*

### 3.2.2 Lập trình xử lý nghiệp vụ

Để tích hợp mã hóa RSA và PBKDF2 vào hệ thống sẽ sử dụng thư viện mã hóa

- jsencrypt.js: https://npmcdn.com/jsencrypt@2.3.0/bin/jsencrypt.js .

- CryptoJS v3.1.2: https://code.google.com/archive/p/crypto-js/

Thêm thư viện vào thư mục: Duplicati\Server\webroot\ngax\scripts\libs

Và cấu hình file index.html: Duplicati\Server\webroot\ngax\index.html

#### a) Module cấu hình hệ thống:

File giao diện: Duplicati\Server\webroot\ngax\templates\settings.html

File Controller:

Duplicati\Server\webroot\nganx\scripts\controllers\SystemSettings\Controller.s

File Backend: Duplicati\Server\WebServer\RESTMethods\ServerSettings.cs

Ở giao diện tiến hành thêm cấu hình RSA:

```
<h2 translate>RSA Public Key</h2>
        <div class="input textarea" ng-show="needRSA">
          <textarea id="rsakeypublic" ng-model="rsaPublicKey"
readonly></textarea>
        </div>
        <h2 translate>RSA Private Key</h2>
        <div class="input textarea" ng-show="needRSAPrivate">
           <textarea  id="rsakeyprivate" ng-
```

```
model="rsaPrivateKey" readonly></textarea>
        </div>
        <div class="keysizes">
            <h4 translate>Select RSA Key Size</h4>
            <select style="float: none" ng-
model="selectedSize" ng-options="size for size in
selectOptions"></select>
        </div>
        <br>
        <div class="buttons" style="float: left;" translate>
           <input type="button"id="generateRSAKey"
class="submit" value="Generate" ng-
click="genRSAKey(selectedSize)">
        </div>
        <br>
        <div class="keysizes">
            <h4 translate>Select Cipher Type</h4>
            <select ng-model="selectedCipher" ng-options="size
for size in selectedCipherType"></select>
```

Tại Conltroler của System Settings sẽ xử lý các logic như biểu đồ tuần tự đã thiết kế:

```
$scope.genRSAKey = function (keySize) {
const rsa = forge.pki.rsa;
const keyPair = rsa.generateKeyPair()
const privateKeyPem =
forge.pki.privateKeyToPem(keyPair.privateKey);
const publicKeyPem =
forge.pki.publicKeyToPem(keyPair.publicKey);
$scope.rsaPublicKey = publicKeyPem;
$scope.rsaPrivateKey = privateKeyPem;
$scope.needRSA = true;
$scope.needRSAPrivate = true;
};
```

Hàm genRSAKey sử dụng thư viện forge tạo ra một cặp khóa bao gồm khóa công khai và khóa bí mật, chuyển khóa riêng tư từ định dạng chuẩn của thư viện sang dạng chuỗi PEM, một đinh dạng văn bản phổ biến để lưu trữ và truyền tải khóa.

Thực hiện mã hóa RSA Private key bằng mật khẩu người dùng:

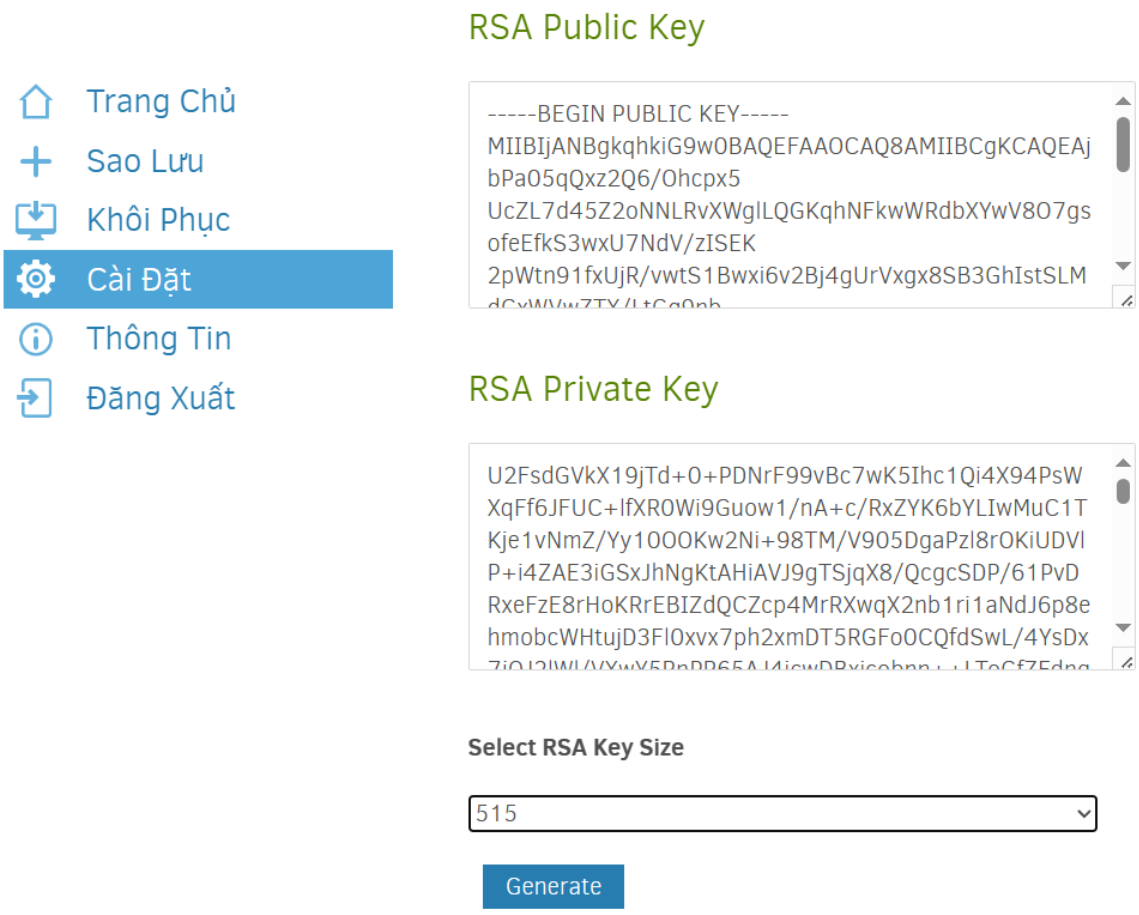
```
$scope.save = function () {
   ……
        var salt = CryptoJS.lib.WordArray.random(128 / 8);
        var saltBase64 = salt.toString(CryptoJS.enc.Base64);
        var key128Bits =
CryptoJS.PBKDF2($scope.remotePassword, salt, {
          keySize: 128 / 32
        });
```

```
var aes_passphrase = key128Bits.toString(CryptoJS.enc.Base64);
var encrypted =
CrytoJS.AES.encrypt($scope.rsaPrivateKey,aes_passphrase);
```

Tạo ra một chuỗi (salt) là một phần quan trọng trong việc răng cường bảo mật cho quá trình sinh khóa. Độ dài của salt là 16 byte để cung cấp cân bằng giữa bảo mật và hiệu suất và chuyển chuỗi salt về chuỗi Base64 để dễ dàng lưu trữ. Sau đó dùng PBKDF2 để tạo một khóa mã hóa 128-bit để tạo ra một khóa AES để mã khóa khóa bí mật RSA được tạo ra trước đó.

Sau khi mã hóa khóa bí mật, phía client truyền xuống phía backend qua phương thức HTTP PATCH đọc dữ liệu và tiến hành lưu vào database

```
 var patchdata = {
        ......
        "rsa-private-key": encrypted,
        "rsa-public-key": $scope.rsaPublicKey,
        "rsa-cipher-type":$scope.selectedCipher,
        "rsa-private-key-salt":saltBase64
      };
  AppService.patch("/serversettings", patchdata, {
        headers: { "Content-Type": "application/json;
charset=utf-8" },
      }).then(function () {
        setUILanguage();
```




*Hình 3.5 Giao diện thiết lập RSA Key*

Sau khi thiết lập trên giao diện thì RSA Key sẽ được mã hóa lưu vào bảng option trong database cấu hình hệ thống

| BackupID | Filter | Name | Value |
|---|---|---|---|
| Filter | Filter | Filter | Filter |
| -2 | | last-webserver-port | 8200 |
| -2 | | is-first-run | |
| -2 | | server-port-changed | True |
| -2 | | server-passphrase | 9Uwr0CUDE8PoUjGPGcYXUL5ZGdLTF... |
| -2 | | server-passphrase-salt | OpFpw+y0d631+d5HBvmqP6ZsYqoj... |
| -2 | | server-passphrase-trayicon | a4d57099-69d3-42b3-93d7-7ad90378... |
| -2 | | server-passphrase-trayicon-hash | vqRRLexJLui/... |
| -2 | | last-update-check | 638499854172484587 |
| -2 | | update-check-interval | |
| -2 | | update-check-latest | |
| -2 | | unacked-error | False |
| -2 | | unacked-warning | False |
| -2 | | server-listen-interface | loopback |
| -2 | | server-ssl-certificate | |
| -2 | | has-fixed-invalid-backup-id | True |
| -2 | | update-channel | |
| -2 | | usage-reporter-level | |
| -2 | | has-asked-for-password-protection | true |
| -2 | | disable-tray-icon-login | false |
| -2 | | allowed-hostnames | |
| -2 | | rsa-private-key | U2FsdGVkX19jTd+0+PDNrF99vBc7w... |
| -2 | | rsa-public-key | -----BEGIN PUBLIC KEY-----... |
| -2 | | rsa-cipher-type | RSA |
| -2 | | rsa-private-key-salt | rGlMCiU+lT702577Ta85OA== |

*Hình 3.6 RSA Key đã được lưu trong database*

**b) Module chạy cấu hình sao lưu**

File giao diện: Duplicati\Server\webroot\ngax\templates\addoredit.html

File Controller:

Duplicati\Server\webroot\ngax\scripts\services\AppUtils.js

Duplicati\Server\webroot\ngax\scripts\controllers\EditBackupController.js

File backend: Duplicati\Server\WebServer\RESTMethods\Backup.cs

Việc tạo key AES và mã hóa key AES sẽ được thực hiện tại Controller:

Sinh khóa AES ngẫu nhiên cho bản sao lưu:

```
this.generatePassphrase = function() {
        var specials = '!@#$%^&*()_+{}:"<>?[];\',./';
        var lowercase = 'abcdefghijklmnopqrstuvwxyz';
        var uppercase = lowercase.toUpperCase();
        var numbers = '0123456789';
        var all = specials + lowercase + uppercase + numbers;

        function choose(str, n) {
            var res = '';
            for (var i = 0; i < n; i++) {
                res += str.charAt(Math.floor(Math.random() *
str.length));
            }
            return res;
        };
        var pwd = (
            choose(specials, 2) +
            choose(lowercase, 2) +
            choose(uppercase, 2) +
            choose(numbers, 2) +
            choose(all, (Math.random()*5) + 5)
        ).split('');

        for(var i = 0; i < pwd.length; i++) {
            var pos = parseInt(Math.random() * pwd.length);
            var t = pwd[i]
            pwd[i] = pwd[pos];
            pwd[pos] = t;
        }
        return pwd.join('');
    }
```

Hàm generatePassphrase() sinh ra mật khẩu với cấu trúc 2 ký tự đặc biệt, 2 ký tự chữ thường, 2 ký tự chữ hoa, 2 ký tự số và một số ký tự ngẫu nhiên từ 5-9 ký tự do (Math.random() 5*5). Sau khi sinh khóa AES cho bản sao lưu, ta sẽ dùng khóa RSA public để mã hóa khóa AES của bản sao lưu.

```
  $scope.save = function () {
      var key = AppUtils.generatePassphrase();
      $scope.PassphraseScore = 3;
      const rsaPublicKey = $scope.rawdata["rsa-public-key"];
      const encrypt = new JSEncrypt();
        encrypt.setPublicKey(rsaPublicKey);
       const hash = encrypt.encrypt(key);
        if(!hash){
            return false;
```

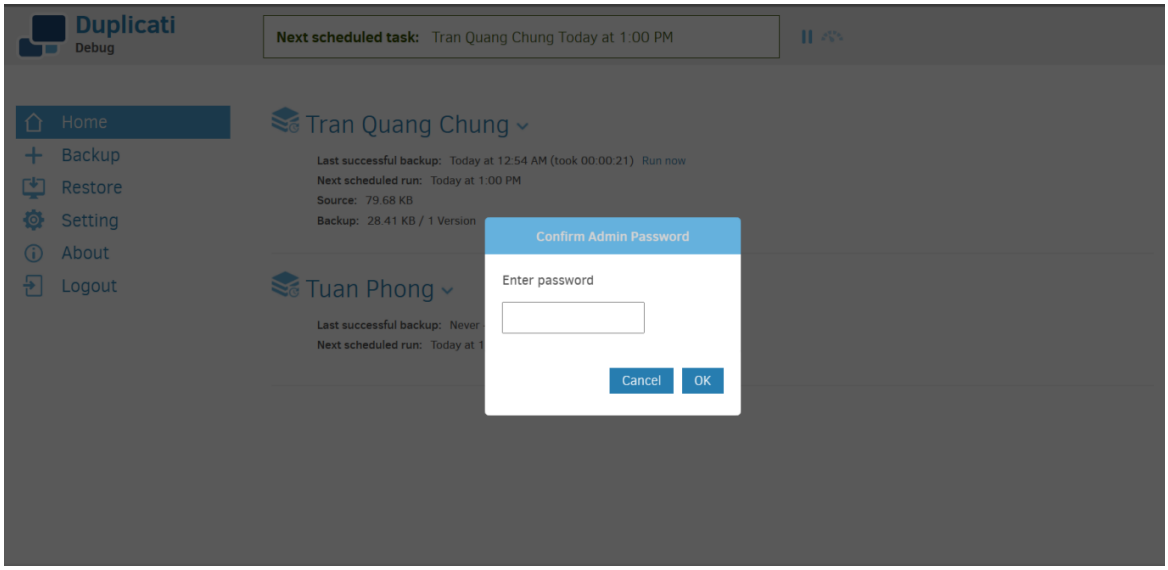
```
            }
            $scope.Options['passphrase'] = hash;
}
```

Kết quả là trong Database cấu hình backup sẽ lưu trữ key AES đã được mã hóa bởi RSA Public key

| BackupID | Filter | Name | Value |
|---|---|---|---|
| Filter | Filter | Filter | Filter |
| -2 | | server-passphrase-salt | OpFpw+y0d631+d5HBvmqP6ZsYqoj... |
| -2 | | server-passphrase-trayicon | 5d712205-8a7a-4275-961c-... |
| -2 | | server-passphrase-trayicon-hash | yeREyG9lDG8qDnUvSHdBI72iK+hNIIr... |
| -2 | | last-update-check | 638499854172484587 |
| -2 | | update-check-interval | |
| -2 | | update-check-latest | |
| -2 | | unacked-error | False |
| -2 | | unacked-warning | False |
| -2 | | server-listen-interface | loopback |
| -2 | | server-ssl-certificate | |
| -2 | | has-fixed-invalid-backup-id | True |
| -2 | | update-channel | |
| -2 | | usage-reporter-level | |
| -2 | | has-asked-for-password-protection | true |
| -2 | | disable-tray-icon-login | false |
| -2 | | allowed-hostnames | |
| -2 | | rsa-private-key | U2FsdGVkX1+OQ/... |
| -2 | | rsa-public-key | -----BEGIN PUBLIC KEY-----... |
| -2 | | rsa-cipher-type | RSA |
| -2 | | rsa-private-key-salt | CX5GfA4UJxRpsAbdU5meaA== |
| 46 | | encryption-module | aes |
| 46 | | compression-module | zip |
| 46 | | dblock-size | 50mb |
| 46 | | passphrase | W9oCnVic5iqLhewsOGjWNd+kLuvPz2... |

*Hình 3.7 Key AES bản sao lưu đã được mã hóa*

Khi thực hiện chạy Backup file, quản trị cần cung cấp Mật khẩu để có thể thực hiện:




*Hình 3.8 Kiểm tra mật khẩu trước khi chạy sao lưu*

Sau khi kiểm tra mật khẩu và so khớp thì ở phía backend sẽ khởi tạo các tiến tình để xử lý quá trình sao lưu file gồm các bước:

```
 Backup.DataBlockProcessor.Run(database, options, taskreader),
 Backup.FileBlockProcessor.Run(snapshot, options, database,
stats, taskreader, token),
 Backup.StreamBlockSplitter.Run(options, database,
taskreader),
 Backup.FileEnumerationProcess.Run(sources, snapshot,
journalService,
     options.FileAttributeFilter, sourcefilter, filter,
options.SymlinkPolicy,
     options.HardlinkPolicy, options.ExcludeEmptyFolders,
options.IgnoreFilenames,
     options.ChangedFilelist, taskreader, token),
 Backup.FilePreFilterProcess.Run(snapshot, options, stats,
database),
 Backup.MetadataPreProcess.Run(snapshot, options, database,
lastfilesetid, token),
 Backup.SpillCollectorProcess.Run(options, database,
taskreader),
 Backup.ProgressHandler.Run(result)
```

Ở đây sẽ tập trung chính vào tiến trình **DataBlockProcessor**

**DataBlockProcessor :** Nhận dữ liệu đầu vào sau đó chia nhỏ thành các khối. Sau khi nhận khối dữ liệu thì sẽ mã hóa chúng, lưu trữ thông tin về các khối bao gồm: vị trí, kích thước và các giá trị băm để kiểm tra tính toàn vẹn

```
blockvolume.AddBlock(b.HashKey, b.Data, b.Offset, (int)b.Size,
b.Hint);
if (indexvolume != null)
{
```

```
    indexvolume.AddBlock(b.HashKey, b.Size);
    if (b.IsBlocklistHashes && fullIndexFiles)
        indexvolume.AddBlockListHash(b.HashKey, b.Size,
b.Data);
}
```

Thêm một khối mới bao gồm dữ liệu băm (b.HashKey), dữ liệu thực của khối (b.Data), vị trí bắt đầu của khối (b.Offset), kích thước của khối (b.Size), Metadata liên quan đến khối dữ liệu(b.Hint)

```
    internal BackendHandler.FileEntryItem
CreateFileEntryForUpload(Options options)
    {
        var fileEntry = new
BackendHandler.FileEntryItem(BackendActionType.Put,
this.RemoteFilename);
        fileEntry.SetLocalfilename(this.LocalFilename);
        fileEntry.Encrypt(options);
        fileEntry.UpdateHashAndSize(options);
        return fileEntry;
    }
```

Mã hóa và tạo một mục nhập cho khối dữ liệu

```
 public override Stream Encrypt(Stream input)
 {
     var cryptoStream = new SharpAESCrypt.SharpAESCrypt(m_key,
input, SharpAESCrypt.OperationMode.Encrypt);
     if (m_usethreadlevel != 0) cryptoStream.MaxCryptoThreads
= m_usethreadlevel;
     return cryptoStream;
 }
```

Quá trình này đảm bảo khi một khối dữ liệu được ghi đầy, nó sẽ đóng và gửi đi một cách an toàn.

### c) Module chạy khôi phục file

Quá trình này sẽ giải dùng RSA private Key để giải mã khóa AES của bản sao lưu và dùng nó để khôi phục lại bản sao lưu ban đầu.

File giao diện: Duplicati\Server\webroot\ngax\templates\restore.html

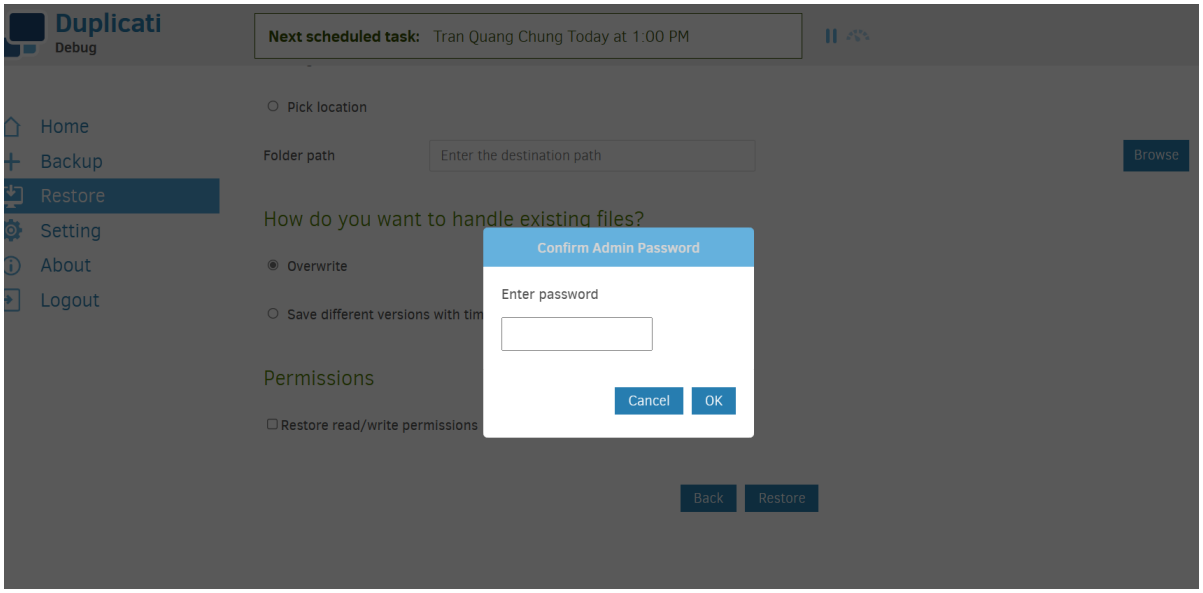
File Controller:

Duplicati\Server\WebServer\RESTMethods\Verifypassword.cs

Duplicati\Server\webroot\ngax\scripts\controllers\RestoreController.js

File Backend: Duplicati\Server\WebServer\RESTMethods\backup.cs

**Giao diện sẽ yêu cầu nhập mật khẩu khi thực hiện Restore file:**




*Hình 3.9 Kiểm tra mật khẩu trước khi chạy khôi phục*

```
 let dataCheck = { password: passInput };
AppService.post("/verifypassword", dataCheck).then(function
(data) {
 if (data.data.result) {
  $scope.passServer = data.data.pwd;
 const salt =
CryptoJS.enc.Base64.parse($scope.rsaPrivateKey_salt);
var key128bits = CryptoJS.PBKDF2($scope.passServer,salt,{
keySize: 128 / 32});
 var aes_passphrase =
key128bits.toString(CryptoJS.enc.Base64);

 var decryptAES = CryptoJS.AES.decrypt(
 $scope.rsaPrivateKey_encrypt,aes_passphrase
).toString(CryptoJS.enc.Utf8);
 $scope.rsaPrivateKey = decryptAES; //result RSA
```

Dùng mật khẩu người dùng và chuỗi salt đã sinh ban đầu để tạo lại khóa AES và giải mã khóa bí mật RSA. Sau khi đã có khóa bí mật RSA sẽ tiến hành giải mã khóa AES của bản sao lưu và truyền xuống cho backend để tiến hành khôi phục.

```
 $scope.onStartRestoreProcess = function () {
     const decrypt = new JSEncrypt();
      decrypt.setPrivateKey($scope.rsaPrivateKey);
      $scope.passphrase  =
decrypt.decrypt($scope.passphrase);
      var p = {….
        passphrase: $scope.passphrase
      };
```

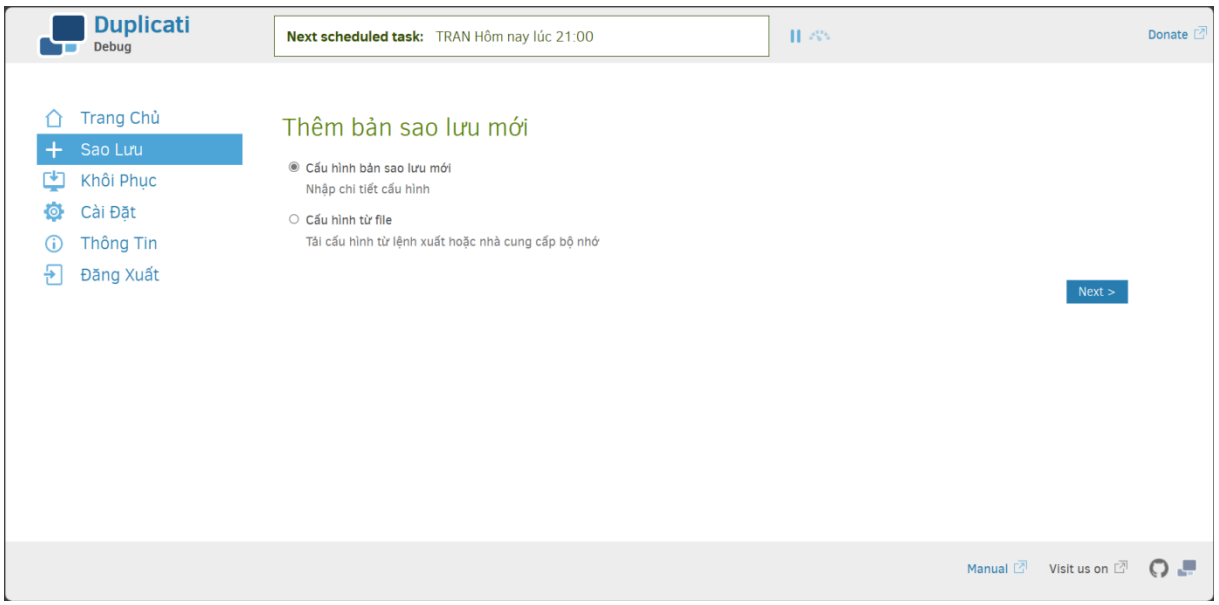
```
    AppService.post("/backup/" + $scope.BackupID +
"/restore", p).then(
          function (resp) {
            $scope.ConnectionProgress =
              gettextCatalog.getString("Restoring files …");
            var t2 = ($scope.taskid = resp.data.TaskID);
            ServerStatus.callWhenTaskCompletes(t2, function
() {
              $scope.onRestoreComplete(t2);
            });
```

Vậy là cơ bản đã tích hợp xong RSA vào Duplicati. Tiếp theo sẽ triển khai hệ thống và dùng thử ứng dụng.

## 3.3. Kiểm thử và đánh giá hệ thống

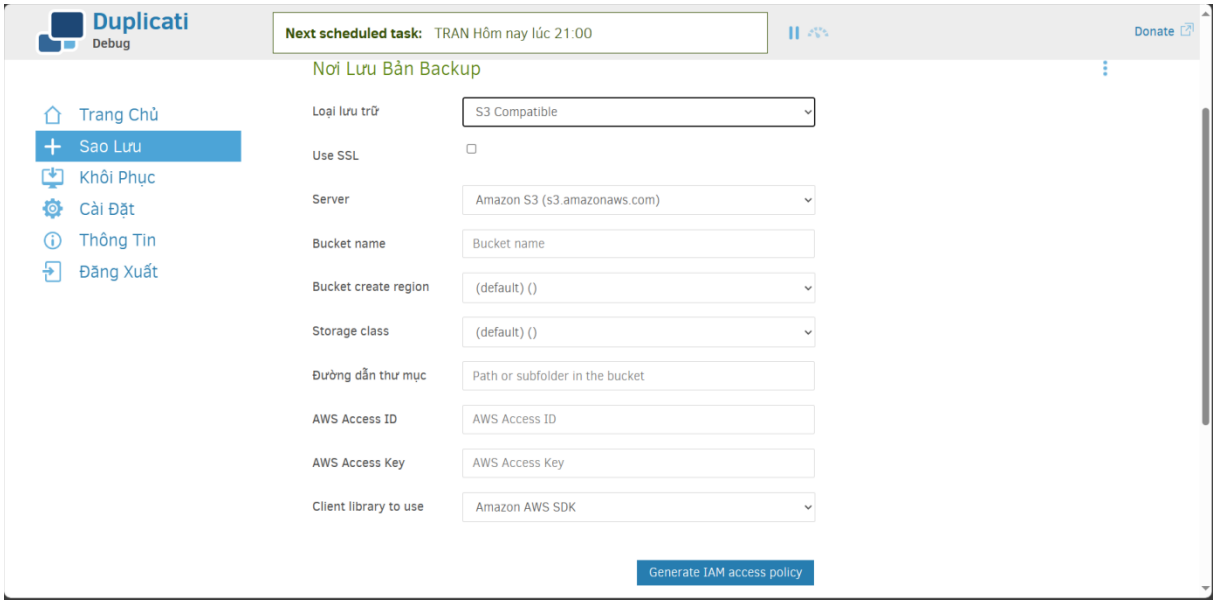
### 3.3.1 Triển khai và kiểm thử tính năng sao lưu dữ liệu

Trên menu chọn Sao Lưu -> Cấu hình bản sao lưu mới




*Hình 3.10 Tạo cấu hình backup mới*

Nhập các thông tin của cơ sở dữ liệu cần sao lưu và vị trí sao lưu trên Amazon S3

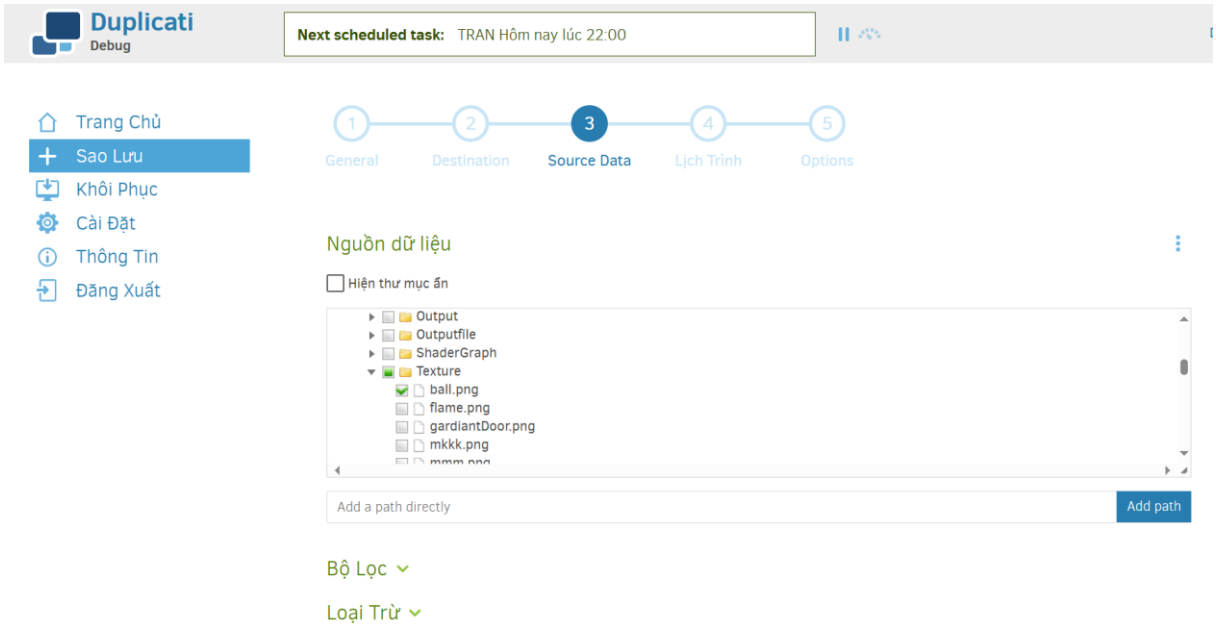



*Hình 3.11 Nhập cấu hình thông tin Amazon S3*

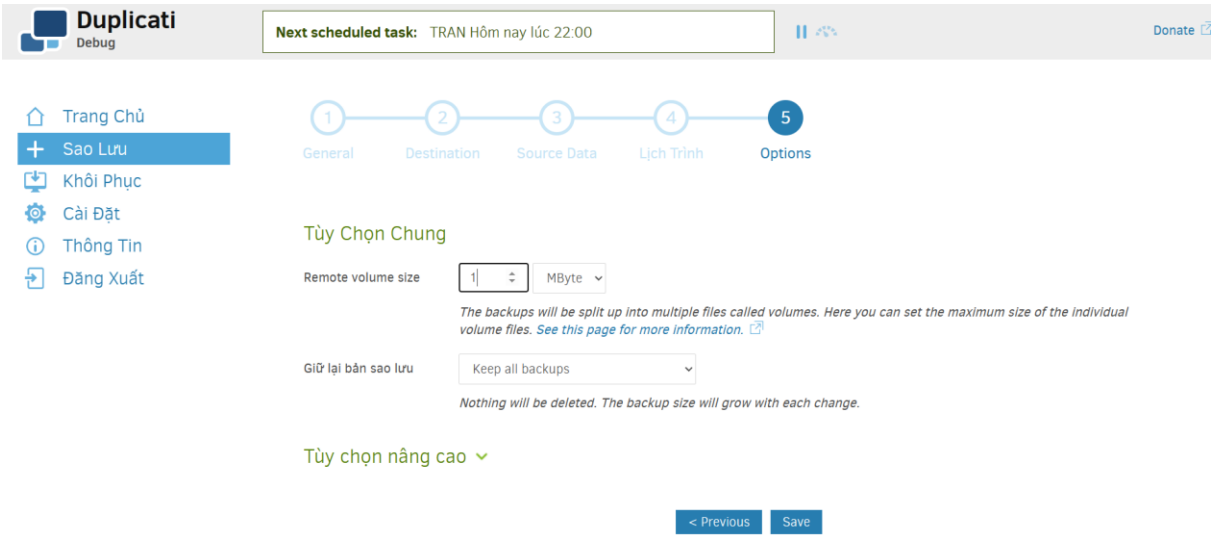
Path on server là thư mục trên Amazon S3 sẽ lưu trữ bản backup

AWS Acesss ID là chuỗi ký tự để xác định người dùng

AWS Acesss Key là chuỗi bí mật dùng để ký các yêu cầu API gửi tới AWS

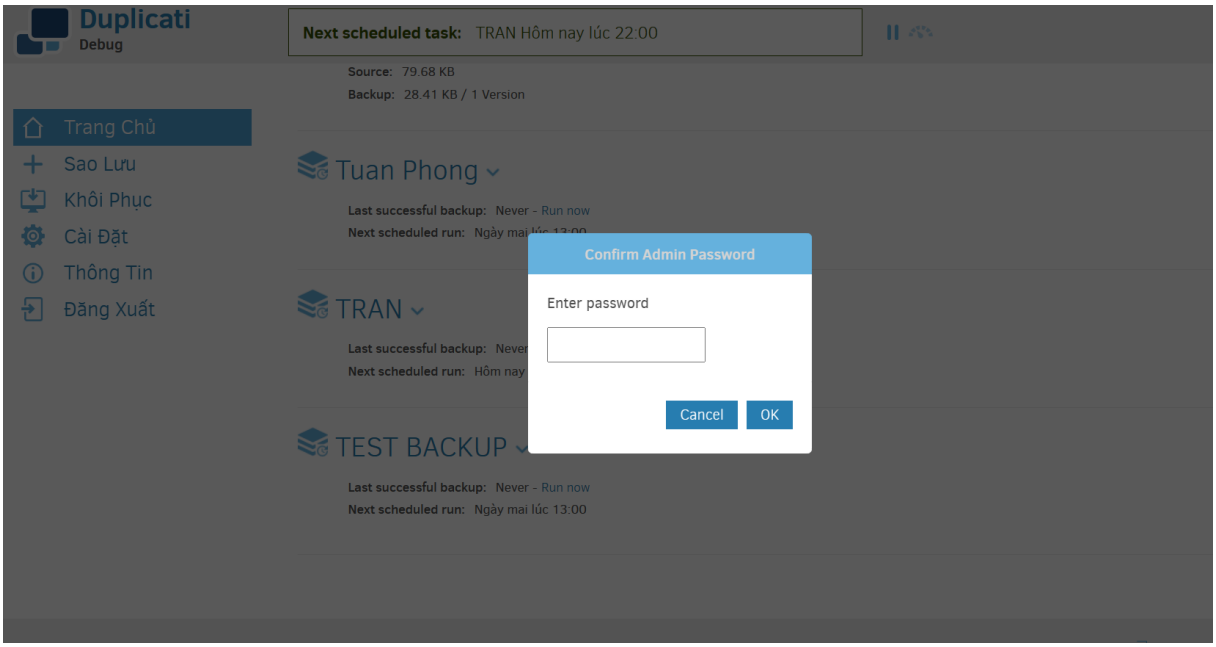



*Hình 3.12 Chọn nguồn dữ liệu cần sao lưu*

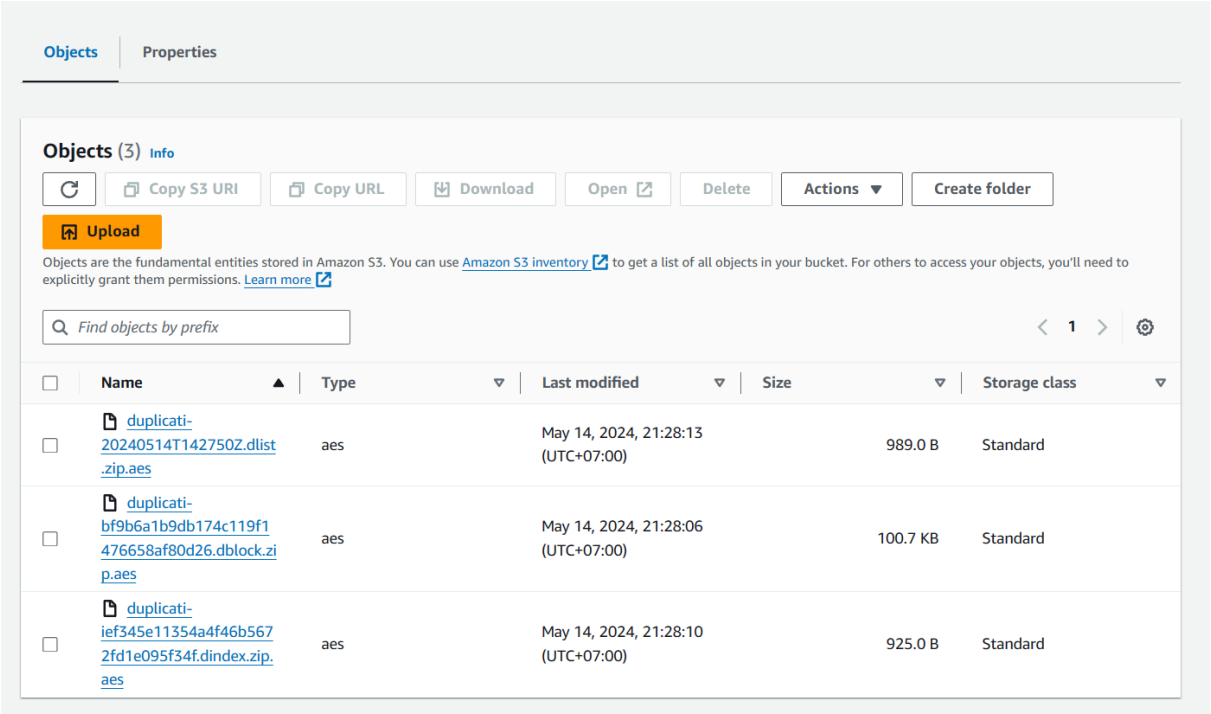



*Hình 3.13 Thiết lập khối dữ liệu mã hóa*

Sau khi thiết lập xong, ta tiến hành nhập mật khẩu để chạy bản sao lưu



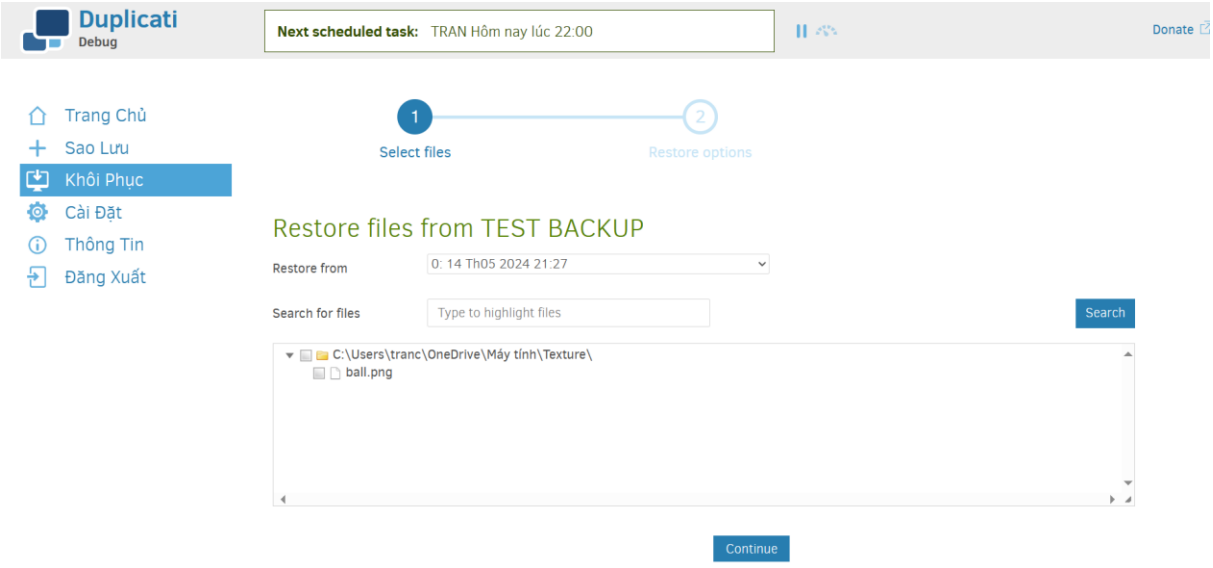

*Hình 3.14 Nhập mật khẩu để chạy bản sao lưu*




*Hình 3.15 File sao lưu đã được mã hóa tải lên Amazon S3*

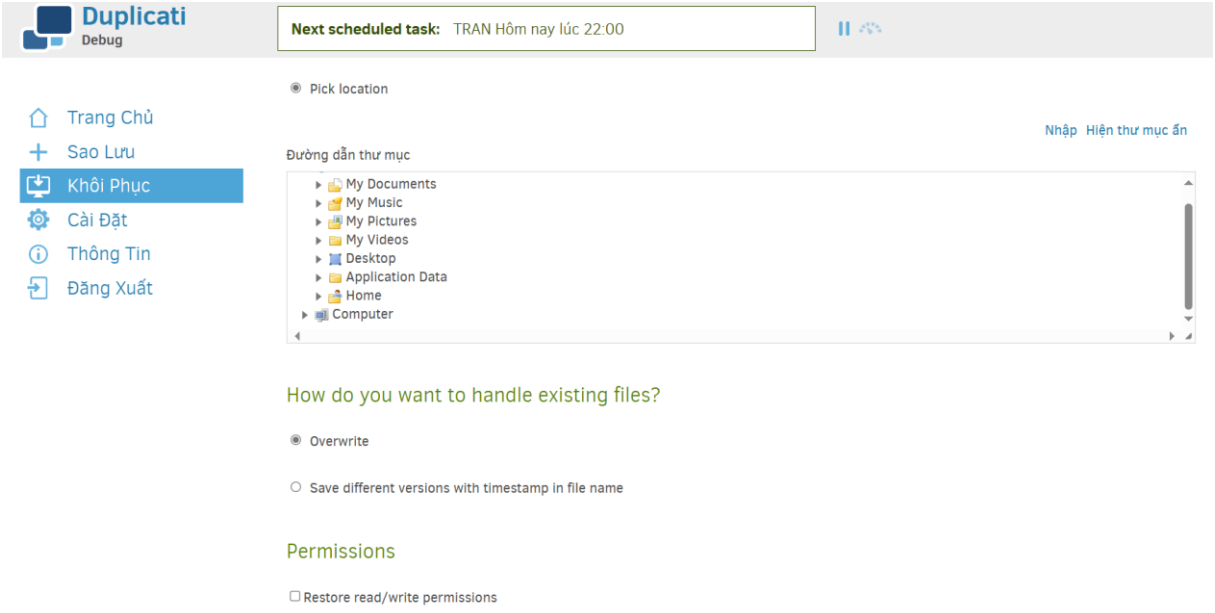
### 3.3.2 Triển khai và kiểm thử tính năng khôi phục dữ liệu

Chọn file muốn khôi phục

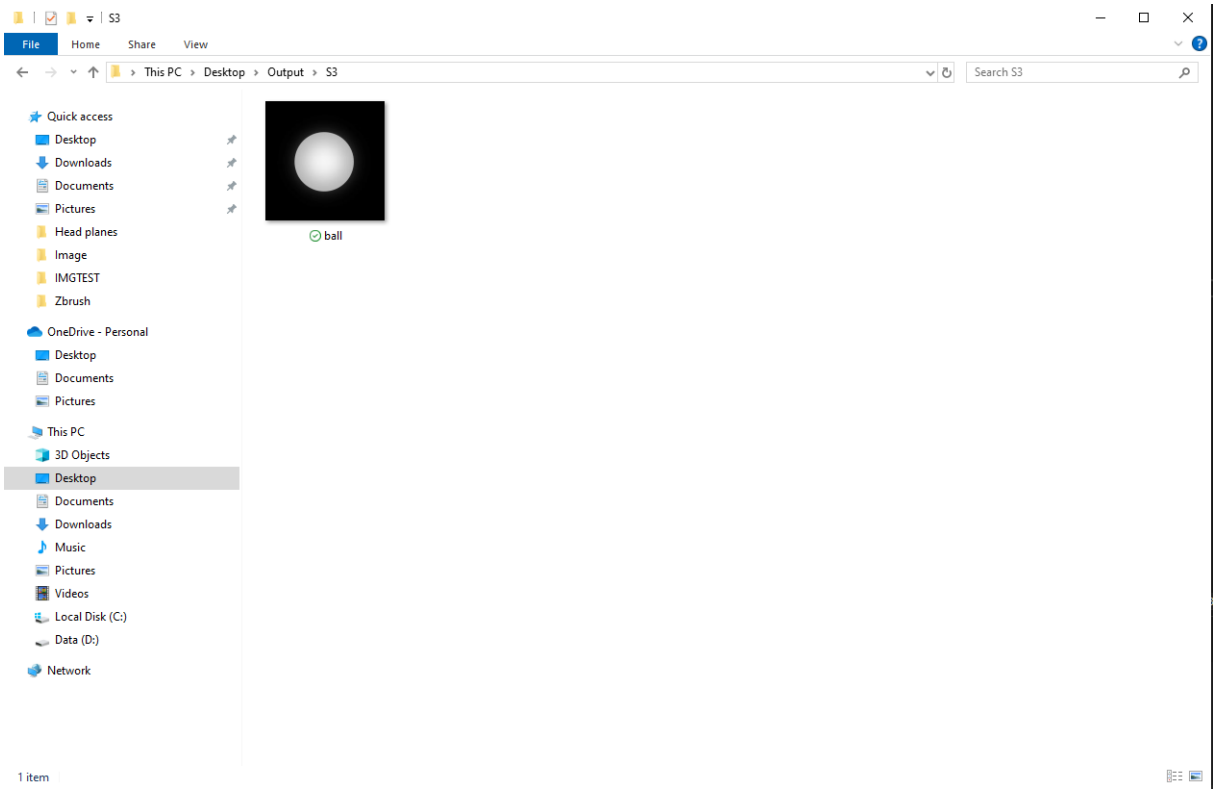



*Hình 3.16 Chọn file muốn khôi phục*

Chọn vị trí lưu file khôi phục




*Hình 3.17 Chọn vị trí file khôi phục sẽ lưu*




*Hình 3.18 Kiểm tra file khôi phục*

### 3.4. Đánh giá hệ thống

Về mặt hiệu suất, hệ thống được thiết kế để đảm bảo quá trình sao lưu và phục hồi dữ liệu diễn ra một cách nhanh chóng và hiệu quả. Để tăng cường bảo mật và độ tin cậy, hệ thống áp dụng mã hóa đầu cuối cho các tệp dữ liệu đã sao lưu, đồng thời tích hợp các cơ chế quản lý quyền truy cập và các biện pháp bảo mật của Amazon S3, nhằm ngăn chặn mất mát dữ liệu và các truy cập không được phép. Nhờ vậy, dữ liệu được sao lưu và phục hồi một cách chính xác và toàn vẹn, đảm bảo hệ thống vận hành ổn định và đúng theo yêu cầu.

Về tính tương thích, hệ thống hiện tại mới chỉ được kiểm thử trên nền tảng Windows và sử dụng hệ quản trị cơ sở dữ liệu SQLite. Chưa có đánh giá về khả năng tương thích của hệ thống với các hệ điều hành hoặc môi trường khác.

Về khả năng mở rộng, hệ thống có thể thích ứng với việc xử lý lượng dữ liệu ngày càng tăng, và có thể được mở rộng theo nhu cầu. Tuy nhiên, việc mở rộng này có thể yêu cầu tăng không gian lưu trữ và cường độ tài nguyên hệ thống, điều này có thể dẫn đến việc tăng chi phí đáng kể.

### 3.5. Tổng kết chương

Trong chương này, đồ án đã tập trung vào việc nghiên cứu, thực hiện và triển khai hệ thống sao lưu và phục hồi dữ liệu dựa trên giao thức S3 và Duplicati. Đồ án đã tiến hành tìm hiểu, phân tích mã nguồn của Duplicati để hiểu cách thức hoạt động và cấu trúc của phần mềm, đặc biệt là các module xử lý và tương tác với các dịch vụ lưu trữ sử dụng giao thức S3 và module mã hóa dữ liệu. Từ đó, đã xây dựng tính năng sao lưu và phục hồi cơ sở dữ liệu sử dụng Amazon S3. Qua việc tiến hành kiểm thử và đánh giá để đảm bảo tính ổn định, hiệu suất và đáng tin cậy của hệ thống, đồ án đã thực hiện kiểm thử các tình huống xử lý dữ liệu khác nhau, đảm bảo rằng hệ thống hoạt động chính xác và đáp ứng các yêu cầu đề ra. Các hoạt động này đóng vai trò quan trọng trong việc đảm bảo tính hoàn thiện và hiệu suất của hệ thống sao lưu và phục hồi dữ liệu.

# KẾT LUẬN

Trong báo cáo này, đồ án đã trình bày về phương pháp của mình về việc sao lưu và phục hồi dữ liệu hiệu quả thông qua sử dụng giao thức S3 cùng với công cụ Duplicati tích hợp mã hóa. Qua quá trình nghiên cứu và phát triển, đã đạt được trong đồ án này bao gồm:

- Nghiên cứu về cách thức hoạt động của các luồng sao lưu và phục hồi dữ liệu
- Nghiên cứu tích hợp mô hình mã hóa đầu cuối vào quá trình sao lưu và khôi phục
- Nghiên cứu về giao thức S3 và dịch vụ Amazon S3
- Tìm hiểu thêm về những công nghệ mới: ngôn ngữ, framework, library … trong quá trình xây dựng và triển khai phần mềm
- Xây dựng thành công Hệ thống sao lưu và khôi phục đáp ứng nhu cầu sử dụng
- Hiểu thêm về design parttern mà hệ thống áp dụng

Trong quá trình nghiên cứu đề tài, còn có một số mặt hạn chế của đồ án, đây cũng là những vấn đề cần phải cải thiện trong tương lai:

- Chưa đánh giá sự tương thích của hệ thống trên các môi trường, hệ điều hành khác.
- Chi phí và tài nguyên: xây dựng và duy trì hệ thống sao lưu và phục hồi dữ liệu trên giao thức S3 có thể đòi hỏi chi phí và tài nguyên đáng kể.

Qua đó, đề xuất một số hướng phát triển khả thi có thể được thực hiện trong tương lai:

- Thử nghiệm ứng dụng trên các hệ quản trị cơ sở dữ liệu khác như MySQL, SQL Server…
- Đánh giá sự tương thích của hệ thống trên các hệ điều hành khác.
- Xây dựng unit test để đảm bảo chất lượng sản phẩm.
- Tăng tính bảo mật và quyền riêng tư, nâng cao độ tin cậy của ứng dụng.

# TÀI LIỆU THAM KHẢO

[1] Acronis (2020), *Incremental vs Differential vs Full Backup - Definition and Difference*.

[2] Angular, *Understanding MVC-Services for Frontend*.

[3] Amazon Web Services, *What is Amazon S3?*

[4] Amazon Web Services, *Policy evaluation logic - AWS Identity and Acess Management*

[5] Duplicati (2018), *How the Backup and Restore Process Works*

[6] Evgeny Milavov (2009), *The RSA Algorithm*.

[7] Flevina JoneseD' Souza (2009), *Advenced Encryption Standard*