BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

KHOA: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: AN TOÀN BẢO MẬT THÔNG TIN

ĐỀ TÀI 8:

NGHIÊN CỨU BẢO MẬT THÔNG TIN MÃ HÓA VÀ GIẢI MÃ VĂN BẢN BẰNG HỆ MÃ HÓA VIGENERE

Sinh viên thực hiện	Lớp	Khóa
Hà Tiến Dũng	DCCNTT12.10.12	K12
Nguyễn Văn Đạt	DCCNTT12.10.12	K12
Trần Thanh Bình	DCCNTT12.10.12	K12
Vũ Thanh Hải	DCCNTT12.10.12	K12

Bắc Ninh, năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

KHOA: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: AN TOÀN BẢO MẬT THÔNG TIN

Nhóm: 8

Đề tài 8:

Nghiên cứu bảo mật thông tin mã hóa và giải mã văn bản bằng hệ mã hóa vigenere

STT	Sinh viên thực hiện	Mã sinh viên	Điểm bằng số	Điểm bằng chữ
1	Hà Tiến Dũng	20213409		
2	Nguyễn Văn Đạt	20213571		
3	Trần Thanh Bình	20214075		
4	Vũ Thanh Hải	20213345		

CÁN BỘ CHẤM 1

(Ký và ghi rõ họ tên)

CÁN BỘ CHẤM 2

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT 1
DANH MỤC HÌNH ẢNH 2
CHƯƠNG 1: MỞ ĐẦU 3
1.1 Giới thiệu đề tài 3
1.1.1 Tên đề tài 3
1.1.2 Nội dung 3
1.1.3 Mục đích 4
1.1.4 Ý nghĩa của đề tài 5
1.2 Công cụ sử dụng 5
CHƯƠNG 2: GIẢI QUYẾT VẤN ĐỀ 9
2.1 Giới thiệu đầu vào và đầu ra của bài toán 9
2.2 Cơ sở lý thuyết 9
2.3 Giới thiệu thuật toán 11
CHƯƠNG 3: GIỚI THIỆU CHƯƠNG TRÌNH 13
3.1 Giới thiệu chương trình 13
3.1.1 Giao diện chương trình 13
3.1.2 Đầu vào và đầu ra 14
3.1.3 Xây dựng chương trình 15
3.2 Kết quả thu được 20
KẾT LUẬN 24
Kết luận 24
Hạn chế và hướng phát triển của đề tài 24
TÀI LIỆU THAM KHẢO 25

DANH MỤC CÁC TỪ VIẾT TẮT

STT	Từ viết tắt	Ý nghĩa
1	IDE	Integrated Development Environment
2	API	Application Programming Interface
	XML	eXtensible Markup Language
	HTML	HyperText Markup Language
	GUI	Graphical User Interface

DANH MỤC HÌNH ẢNH

Hình 3.1 Giao diện chính của chương trình. 13
Hình 3.2 Giao diện tệp hợp lệ để mã hóa 14
Hình 3.3 Tiến hành chọn file để mã hóa.....20
Hình 3.4 Kết quả sau khi chọn file 21
Hình 3.5 kết quả khi nhập mã khóa và ấn nút encrypt để mã hóa.....21
Hình 3.6 Tiến hành lưu file kết quả mã hóa. 22
Hình 3.7 Kết quả sau khi lưu file.....22
Hình 3.8 Tiến hành chọn file để giải mã. 23
Hình 3.9 Kết quả giải mã sau khi nhập mã khóa và ấn nút decrytion. 23

CHƯƠNG 1: MỞ ĐẦU

1.1 Giới thiệu đề tài

1.1.1 Tên đề tài

Đề tài "Nghiên cứu bảo mật thông tin mã hóa và giải mã văn bản bằng hệ mã hóa Vigenere" tập trung vào việc nghiên cứu về phương pháp mã hóa và giải mã thông tin sử dụng hệ mã hóa Vigenere, đồng thời tập trung vào các vấn đề liên quan đến bảo mật thông tin.y

1.1.2 Nội dung

Trong phạm vi đề tài này, nội dung chủ yếu được tập trung vào các phương pháp mã hóa, giải mã vã khả năng bảo mật thông tin bằng hệ mã hóa Vigenere .

Phương pháp mã hóa thông tin văn bản bằng hệ mã hóa Vigenere được sử dụng để bảo vệ thông tin trong quá trình truyền tải và lưu trữ. Dưới đây là một số mục đích cụ thể mà phương pháp này được áp dụng:

Bảo mật thông tin: Mã hóa Vigenere giúp bảo vệ thông tin khỏi việc bị đánh cắp hoặc đọc trộm trong quá trình truyền tải. Khi thông tin được mã hóa, người nhận cần phải có khóa để giải mã để có thể đọc được nội dung ban đầu.

Gửi tin nhắn bí mật: Trong quân sự, ngoại giao, và các lĩnh vực y tế, việc gửi tin nhắn mà không muốn bị tiết lộ cho bên thứ ba là rất quan trọng. Mã hóa Vigenere cung cấp một cách thức đơn giản để gửi tin nhắn bí mật mà chỉ có người nhận có khóa mới có thể giải mã.

Bảo vệ thông tin cá nhân: Trong môi trường trực tuyến, việc bảo vệ thông tin cá nhân và tài khoản trở nên ngày càng quan trọng. Mã hóa Vigenere có thể được sử dụng để mã hóa các thông tin như mật khẩu, số thẻ tín dụng, hay thông tin cá nhân khác trước khi lưu trữ hoặc truyền tải.

Giữ bí mật trong kinh doanh và thương mại: Trong môi trường kinh doanh và thương mại, có thể có nhiều thông tin nhạy cảm cần được bảo vệ khỏi việc tiết lộ cho đối thủ cạnh tranh hoặc công chúng. Mã hóa Vigenere có thể được sử dụng để bảo vệ các thông tin như kế hoạch kinh doanh, dữ liệu khách hàng, hay thông tin sản phẩm mới.

Phương pháp giải mã thông tin văn bản bằng hệ mã hóa Vigenere được sử dụng để khôi phục lại thông tin gốc từ một văn bản đã được mã hóa bằng phương pháp này. Dưới đây là một số mục đích cụ thể của việc giải mã bằng hệ mã hóa Vigenere:

Đọc tin nhắn được mã hóa: Khi nhận được một tin nhắn đã được mã hóa bằng Vigenere, việc giải mã là cần thiết để có thể đọc được nội dung của tin nhắn. Người nhận cần phải biết khóa để giải mã tin nhắn thành thông tin ban đầu.

Phân tích thông tin quan trọng: Trong nhiều trường hợp, thông tin được mã hóa bằng Vigenere có thể chứa các thông tin quan trọng, như kế hoạch kinh doanh, thông tin tài chính, hay thông tin quân sự. Việc giải mã giúp người nhận nắm bắt được thông tin này và có thể thực hiện các biện pháp hoặc quyết định dựa trên nội dung đã được khôi phục.

Phân tích dữ liệu nghiên cứu: Trong lĩnh vực nghiên cứu, có thể có những dữ liệu quan trọng đã được mã hóa bằng Vigenere để bảo vệ khỏi việc tiết lộ trước khi công bố. Việc giải mã dữ liệu này giúp các nhà nghiên cứu có thể phân tích và hiểu sâu hơn về thông tin đã được ẩn trong dữ liệu.

Bảo vệ quyền riêng tư: Trong một số trường hợp, việc giải mã dữ liệu bằng Vigenere cũng có thể được sử dụng để bảo vệ quyền riêng tư của cá nhân hoặc tổ chức. Người nhận

có thể sử dụng việc giải mã để đảm bảo rằng chỉ những người được ủy quyền mới có thể truy cập vào thông tin.

Khả năng bảo mật thông tin mang lại khả năng bảo mật thông tin tốt với nhiều ưu điểm nổi bật :

Do sử dụng nhiều bảng chữ cái thay thế xen kẽ nhau, Vigenere tạo ra chuỗi văn bản mã hóa phức tạp, ít lặp lại, khiến việc phân tích tần suất chữ cái trở nên khó khăn hơn, cản trở nỗ lực bẻ khóa của kẻ tấn công.

Vigenere có khả năng chống lại các phương pháp tấn công phân tích mật mã phổ biến như phân tích tần suất, phân tích cặp chữ cái do thiếu đi các mẫu lặp lại rõ ràng trong văn bản mã hóa.

Thuật toán mã hóa và giải mã Vigenere tương đối đơn giản, dễ dàng triển khai và sử dụng mà không cần kiến thức toán học cao siêu.

Đi kèm với nhưng ưu điểm kể trên thì hệ mã hóa Vigenere cũng tồn tại một số hạn chế như:

Độ an toàn phụ thuộc vào độ dài và độ bí mật của từ khóa: Hiệu quả bảo mật của Vigenere phụ thuộc chủ yếu vào độ dài và tính bí mật của từ khóa. Một từ khóa ngắn hoặc dễ đoán có thể khiến hệ thống dễ bị bẻ khóa hơn.

Yêu cầu truyền tải từ khóa an toàn: Việc trao đổi từ khóa an toàn giữa người gửi và người nhận là một vấn đề quan trọng. Nếu kẻ tấn công có thể lấy được từ khóa, họ có thể dễ dàng giải mã tin nhắn.

Yếu điểm đối với các ngôn ngữ có tần suất chữ cái không đồng đều: Vigenre có thể kém hiệu quả hơn đối với các ngôn ngữ có tần suất chữ cái không đồng đều, do khả năng phân tích tần suất chữ cái có thể cao hơn.

1.1.3 Mục đích

Nghiên cứu về hệ mã hóa Vigenere: Hiểu rõ về nguyên lý hoạt động, cơ chế mã hóa và giải mã của hệ mã hóa Vigenere.

Đánh giá độ an toàn của Vigenere: Phân tích độ an toàn của phương pháp mã hóa và giải mã này trước các phương pháp tấn công thông tin.

Áp dụng vào thực tiễn: Nghiên cứu cách áp dụng hệ mã hóa Vigenere vào các tình huống bảo mật thông tin cụ thể, như bảo vệ thông tin cá nhân, truyền dữ liệu an toàn, hoặc trong các hệ thống an ninh mạng.

Nâng cao nhận thức về bảo mật thông tin: Tạo ra một đề tài nghiên cứu có thể giúp tăng cường nhận thức và hiểu biết về các phương pháp bảo mật thông tin, đồng thời khám phá và thảo luận về những thách thức và cơ hội trong lĩnh vực này.

Đóng góp vào lĩnh vực nghiên cứu và phát triển: Đề tài này có thể cung cấp thông tin và kiến thức hữu ích cho cộng đồng nghiên cứu và phát triển trong lĩnh vực bảo mật thông tin và hệ mã hóa, khuyến khích sự phát triển và nghiên cứu tiếp theo.

1.1.4 Ý nghĩa của đề tài

Nâng cao nhận thức về bảo mật thông tin: Đề tài giúp cung cấp kiến thức về phương pháp mã hóa và giải mã thông tin, từ đó tăng cường nhận thức về bảo mật thông tin trong cộng đồng.

Hiểu biết sâu rộng về hệ mã hóa Vigenere: Nghiên cứu này cung cấp thông tin chi tiết và hiểu biết sâu sắc về cách hoạt động, đặc điểm và tính an toàn của hệ mã hóa Vigenere.

Đóng góp vào lĩnh vực nghiên cứu và phát triển bảo mật thông tin: Kết quả của đề tài có thể được sử dụng như một tài liệu tham khảo cho các nhà nghiên cứu và chuyên gia bảo mật thông tin trong việc phát triển và nghiên cứu các phương pháp mã hóa và giải mã.

Hỗ trợ quyết định trong thực tiễn: Kiến thức từ đề tài có thể giúp người dùng và các tổ chức có cái nhìn rõ ràng hơn về việc lựa chọn và triển khai các biện pháp bảo mật thông tin phù hợp, bao gồm cả việc sử dụng hệ mã hóa Vigenère.

Khuyến khích nghiên cứu tiếp theo: Kết quả từ đề tài này có thể khuyến khích sự quan tâm và nghiên cứu tiếp theo về bảo mật thông tin và các phương pháp mã hóa và giải mã khác, từ đó mở ra những hướng đi mới trong lĩnh vực này.

1.2 Công cụ sử dụng

*** Ứng dụng lập trình: Microsoft Visual Studio 2019**

Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Microsoft Visual Studio còn được gọi là "Trình soạn thảo mã nhiều người sử dụng nhất thế giới ", được dùng để lập trình C++ và C# là chính. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Nó chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (như Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.

Visual Studio hỗ trợ nhiều ngôn ngữ lập trình khác nhau và cho phép trình biên tập mã và gỡ lỗi để hỗ trợ (mức độ khác nhau) hầu như mọi ngôn ngữ lập trình. Các ngôn ngữ tích hợp gồm có C, C++ và C++/CLI (thông qua Visual C++), VB.NET (thông qua Visual

Basic.NET), C# (thông qua Visual C#) và F# (như của Visual Studio 2010). Hỗ trợ cho các ngôn ngữ khác như J++/J#, Python và Ruby thông qua dịch vụ cài đặt riêng rẽ. Nó cũng hỗ trợ XML, HTML, JavaScript và CSS.

*** Ngôn ngữ lập trình: C#**

C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java. C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.

C# là một ngôn ngữ đơn giản: C# loại bỏ một vài sự phức tạp và rối rắm của những ngôn ngữ như Java và c++, bao gồm việc loại bỏ những macro, những template, đa kế thừa, và lớp cơ sở ảo (virtual base class).

Ngôn ngữ C# đơn giản vì nó dựa trên nền tảng C và C++. Nếu chúng ta thân thiện với C và C++ hoặc thậm chí là Java, chúng ta sẽ thấy C# khá giống về diện mạo, cú pháp, biểu thức, toán tử và những chức năng khác được lấy trực tiếp từ ngôn ngữ C và C++, nhưng nó đã được cải tiến để làm cho ngôn ngữ đơn giản hơn.

C# là một ngôn ngữ ít từ khóa: C# là ngôn ngữ sử dụng giới hạn những từ khóa. Phần lớn các từ khóa được sử dụng để mô tả thông tin. Chúng ta có thể nghĩ rằng một ngôn ngữ có nhiều từ khóa thì sẽ mạnh hơn. Điều này không phải sự thật, ít nhất là trong trường hợp ngôn ngữ C#, chúng ta có thể tìm thấy rằng ngôn ngữ này có thể được sử dụng để làm bất cứ nhiệm vụ nào.

Ưu điểm của C#:

+ Dễ học và sử dụng: C# là một ngôn ngữ lập trình dễ học và sử dụng, đặc biệt là cho những người mới bắt đầu. Cú pháp của nó tương đối gọn nhẹ và rõ ràng.

+ Tích hợp tốt với .NET Framework: C# được phát triển bởi Microsoft và tích hợp chặt chẽ với .NET Framework, cung cấp nhiều thư viện và công cụ hỗ trợ mạnh mẽ cho việc phát triển ứng dụng.

+ Bảo mật cao: C# hỗ trợ các tính năng bảo mật như kiểm soát truy cập và quản lý bộ nhớ tự động, giúp giảm thiểu nguy cơ lỗi bảo mật như tràn bộ đệm và injection.

+ Đa nền tảng (Cross-platform): C# có thể được sử dụng để phát triển ứng dụng đa nền tảng thông qua .NET Core và .NET 5, cho phép triển khai ứng dụng trên nhiều hệ điều hành như Windows, macOS và Linux.

Nhược điểm của C#:

+ Hiệu suất không cao như các ngôn ngữ gần sát phần cứng: Mặc dù hiệu suất của C# đã được cải thiện qua các phiên bản .NET mới nhưng vẫn không thể sánh kịp với các ngôn ngữ như C hoặc C++ trong một số trường hợp đặc biệt.

+ Phụ thuộc vào .NET Framework: Mặc dù .NET Framework là một bộ công cụ mạnh mẽ, nhưng phụ thuộc quá nhiều vào nó cũng có thể tạo ra một số vấn đề. Đặc biệt là khi cần triển khai ứng dụng trên các nền tảng không hỗ trợ .NET Framework.

+ Quản lý bộ nhớ do Garbage Collector (GC): C# sử dụng GC để tự động quản lý bộ nhớ, điều này có thể dẫn đến hiện tượng phát sinh overhead và nguy cơ "stuttering" khi GC hoạt động. Tuy nhiên, điều này đã được cải thiện qua các phiên bản .NET mới.

+ Khả năng tùy chỉnh hạn chế: So với các ngôn ngữ lập trình khác như C++ hay Rust, C# có khả năng tùy chỉnh và kiểm soát thấp hơn đối với một số yếu tố như quản lý bộ nhớ và tối ưu hóa mã máy.

*** Thư viện lớp đồ họa Winform**

Winform là một thư viện lớp đồ họa, mã nguồn mở và được cung cấp hoàn toàn miễn phí. Phần mềm này cung cấp nền tảng giúp bạn viết những lập trình đa dạng cho các thiết bị như máy tính bàn, laptop, máy tính bảng,... Winform cũng được coi như là một sự thay thế đối với thư viện lớp nền tảng Microsoft Foundation của C++.

Mỗi màn hình Windows lại cung cấp một giao diện để người dùng có thể giao tiếp với ứng dụng được gọi là GUI (giao diện đồ hoạ của ứng dụng). Nó bao gồm các ứng dụng chạy trên máy tính Windows như Microsoft, Word, Excel, Mail, Access, Yahoo, Calculator,...

Winform có các thành phần cơ bản như Forms Panel, Button Textbox, ComboBox, RadioButton,... Trong đó, Form là nơi chứa tất cả thành phần của chương trình, Panel chứa Button, Label, TextBox. Button là nút nhấn, Textbox dùng để nhập văn bản một dòng hay nhiều dòng. Label hiển thị văn bản hoặc thông tin trên Form và ComboBox là các lựa chọn có sẵn để bạn lựa chọn dễ dàng hơn.

Ưu điểm của WinForms:

+ Dễ học và sử dụng: WinForm cung cấp một mô hình lập trình đơn giản và dễ hiểu, làm cho việc học và sử dụng nhanh chóng đối với những người mới bắt đầu.

+ Tích hợp tốt với Visual Studio: WinForm được tích hợp một cách tốt với Visual Studio, môi trường phát triển phổ biến của Microsoft, giúp tăng hiệu suất làm việc.

+ Hỗ trợ tốt cho kiến trúc hướng sự kiện (event-driven): WinForm hỗ trợ mô hình lập trình hướng sự kiện, giúp dễ dàng xử lý các sự kiện và phản ứng với hành động của người dùng.

+ Hỗ trợ tiếng môi trường .NET rộng rãi: WinForm là một phần của .NET Framework, do đó nó tận dụng được các lợi ích của .NET như tích hợp dữ liệu, bảo mật, và hỗ trợ ngôn ngữ đa dạng.

Nhược điểm của WinForm:

+ Giao diện lỗi thời: WinForm không cung cấp các công cụ thiết kế giao diện hiện đại như WPF (Windows Presentation Foundation), điều này có thể làm cho giao diện của ứng dụng có vẻ lỗi thời so với các ứng dụng mới.

+ Khả năng tùy biến hạn chế: So với các công nghệ mới như WPF hoặc WinUI, WinForm có khả năng tùy biến và mở rộng hạn chế hơn, điều này có thể làm cho việc phát triển ứng dụng phức tạp trở nên khó khăn hơn.

+ Hiệu suất không tốt với giao diện phức tạp: WinForm có thể gặp vấn đề về hiệu suất khi xử lý giao diện với các form hoặc control phức tạp, điều này có thể dẫn đến hiện tượng lag hoặc trễ trong ứng dụng.

+ Giới hạn đa nhiệm và đa luồng: WinForm không hỗ trợ đa nhiệm và đa luồng một cách tự nhiên như các công nghệ khác, điều này có thể gây ra các vấn đề về hiệu suất và khó khăn trong việc xử lý các tác vụ đồng thời trong ứng dụng

CHƯƠNG 2: GIẢI QUYẾT VẤN ĐỀ

2.1 Giới thiệu đầu vào và đầu ra của bài toán

* Hệ mã hóa vigenere là một phương pháp mã hóa văn bản bằng cách sử dụng một bảng chữ cái đặt biệt.

- Đầu vào của bài toán mã hóa và giải mã văn bản bằng hệ mã hóa vigenere bao gồm:

+ Key: Đây là chuỗi ký tự được sử dụng để mã hóa và giải mã văn bản. Khóa có thể có độ dài bất kỳ và được sử dụng lặp đi lặp lại cho đến khi đủ độ dài với văn bản cần mã hóa hay giải mã.

+ Đối với mã hóa đầu vào tiếp theo là một file chứa văn bản cần mã hóa: Đây là văn bản mà người dùng muốn mã hóa.

+ Đối với giải mã đầu vào tiếp theo là một file chứa văn bản đã được mã hóa và thông tin khóa đã mã hóa văn bản đó.

+ Ngoài ra, người dùng cũng có thể nhập tay một chuỗi văn bản mà người dùng muốn mã hóa hoặc giải mã mà không bắt buộc phải chọn file văn bản nào đó.

- Đầu ra của quá trình mã hóa và giải mã bao gồm:

+ Đối với mã hóa là một văn bản đã được mã hóa: Đây là văn bản đã được mã hóa từ văn bản gốc và khóa mà người dùng đã chọn. Người dùng có thể lưu file văn bản lại và file được lưu sẽ có tên + với khóa mà người dùng sử dụng để mã hóa.

+ Đối với giải mã là một văn bản đã được giải mã: Đây là văn bản đã được giải mã từ văn bản đã được mã hóa và khóa mà người dùng đã chọn. Người dùng có thể lưu file văn bản lại và file được lưu sẽ có tên + với khóa mà người dùng sử dụng để giải mã.

2.2 Cơ sở lý thuyết

- Hệ mã hóa Vigenère là một phương pháp mã hóa được sử dụng trong mật mã học. Nó được phát triển bởi Blaise de Vigenère vào thế kỷ 16 và đã được sử dụng rộng rãi trong quân sự và ngoại giao cho đến khi các phương pháp mã hóa hiện đại phát triển.

- Cơ sở lý thuyết của hệ mã hóa Vigenère dựa trên việc sử dụng một bảng Vigenère, là một lưới chữ cái được tạo thành bằng cách xếp chồng lên nhau các bảng chữ cái Caesar (mỗi bảng chứa các chữ cái được dịch chuyển một số bước cố định). Mỗi dòng của bảng Vigenère tương ứng với một khóa mã, và mỗi cột tương ứng với một chữ cái trong văn bản cần mã hóa.

- Quá trình mã hóa bằng hệ mã hóa Vigenère được thực hiện bằng cách dịch chuyển các chữ cái của văn bản gốc theo các giá trị trong khóa mã. Khóa mã được lặp đi lặp lại cho đến khi nó có độ dài bằng văn bản cần mã hóa. Mỗi chữ cái trong văn bản gốc được dịch

chuyển bằng cách sử dụng dòng tương ứng trong bảng Vigenère dựa trên chữ cái tương ứng trong khóa mã.

- Là mã thế đa bảng đơn giản nhất, hiệu quả như dùng nhiều mã Caesar cùng một lúc

- Khóa là một dãy các ký tự có độ dài K = K1K2K3…Kd.

- Chữ thứ i chỉ định dùng bảng chữ thứ i với tịnh tiến tương ứng kí tự Ki.

- Chia bản rõ thành các đối tượng d ký tự.

- Lặp lại từ đầu sau d kí tự của bản rõ, giải mã đơn giản là làm việc ngược lại.

- Ví dụ:

+ Key: deceptivedeceptivedeceptive

+ Plaintext: wearediscoveredsaveyourself

+ Ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ.

+ Với Plaintext: P = P0, P1,…, Pn-1.

+ Với Key: K = K0,K1,…,Km-1 (m<n).

+ Với Ciphertext: C = C0,C1,…,Cn-1.

+ C = C0,C1,C2,…,Cn-1 = E(K,P) = E[(K0,K1,K2,K3,…,Km-1),(P0,P1,P2,P3,…Pn-1)] = (P0 + K0)mod26, (P1 + K1)mod26,…, (Pm-1 + Km-1)mod26, (Pm + K0)mod26, (Pm+1 + K1)mod26,…, (P2m-1 + Km-1)mod26,…

=> Mã hóa : Ci = (Pi + Ki.mod(m))mod26.

=> Giải mã : Ci = (Pi - Ki.mod(m))mod26.

- An toàn của mã Vigenere:

+ Ưu điểm :

- Có chữ cái khác nhau cho cùng chữ của bản rõ.
- Suy ra tần suất của các chữ cái khá đều.

+ Yếu điểm:

- Độ dài mã có hạn nên có thể tạo chu kỳ vòng lặp => Xác định số bảng chữ và lần tìm từng chữ.
- Bắt đầu từ tần suất của chữ => Mã đơn bảng hay mã đa bảng.

- Thám mã Vigenere: Đề xuất bởi Babbage (sau hàng thế kỷ):

+ Xác định độ dài khóa:

- Tìm chuỗi được lặp trong bản mã
- Độ dài khoảng lặp là bội số của chiều dài khóa.

=> Nếu có nhiều khoảng lặp sẽ dễ tìm được độ dài khóa.

- Giả sử độ dài khóa là m => bản mã sử dụng m bảng mã đơn.
- Các ký tự ở các vị trí 1, m+1, 2m+1...trong bản mã sẽ dung chung 1 bảng mã đơn.
- Dùng tần suất xuất hiện ký tự trong văn bản để thám mã trên từng bảng mã đơn.

- Mã khóa tự động – Autokey Cipher:

+ Lý tưởng có khoá dài như bản tin. Vigenere đề xuất khoá tự động với từ khoá được nối vào đầu bản tin tạo thành khoá biết từ khoá có thể khôi phục được một số chữ ban đầu tiếp tục sử dụng chúng cho văn bản còn lại nhưng vẫn còn đặc trưng tần suất để tấn công.

+ Ví dụ: cho từ khoá deceptive:

- key: deceptivewearediscoveredsav.
- plaintext: wearediscoveredsaveyourself.
- ciphertext:ZICVTWQNGKZEIIGASXSTSLVVWLA.

2.3 Giới thiệu thuật toán

- Thuật toán mã hóa vigenere là một phương pháp mã hóa văn bản dữ trên việc sử dụng nhiều phép mã hóa caser khác nhau, dựa trên chữ cái của một từ khóa.

- Để mã hóa, ta sử dụng một bảng chữ cái tiếng anh từ A – Z tương đương từ 0 – 26. Trong quá trình mã hóa, tùy theo mã khóa mà ta sẽ có kết quả khác nhau của văn bản cần mã khóa.

- Tạo khóa lặp người dùng nhập vào một từ khóa bằng tiếng anh và không được có số hay kí tự khác chữ cái và sẽ cho mã khóa lặp lại nhiều lần đến khi số chữ cái của mã khóa bằng số chữ cái của đoạn văn bản cần được mã hóa hoặc giải mã.

- Mã hóa văn bản :

+ Chữ cái đầu tiên của văn bản được mã hóa sẽ bắt đầu với chữ cái đầu tiên của mã khóa. Nó sẽ được mã hóa thành chữ cái trên dòng tương ứng.

+ Tương tự chữ cái thứ hai của văn bản cũng sẽ được mã hóa với chữ cái thứ 2 của từ khóa.

+ Công thức để mã hóa (văn bản gốc [i] + mã khóa [i]) % 26. Ta được kết quả của số chữ cái tương ứng. Nhưng khi làm việc với máy tính để máy tính hiểu được ta cần chuyển chữ cái đã được mã hóa sang chữ cái của bảng mã ASCII bằng cách lấy kết quả + 65 (là ký tự A).

+ Ví dụ văn bản gốc [i] = S và mã khóa [i] = E ⇔ (18 + 4) % 26 = 22. Tương đương chữ cái W sau đó chuyển sang bảng mã ascii để hiển thị lên máy tính 22 + 65 = 87. Ta thấy 87 là chữ cái W của bảng mã ascii.

- Giải mã văn bản:

+ Để giải mã văn bản ta sử dụng các ngược lại với quá trình mã hóa

+ Công thức để giải mã (văn bản đã được mã hóa [i] - mã khóa [i]) % 26. Ta được kết quả của số chữ cái tương ứng. Nhưng khi làm việc với máy tính để máy tính hiểu được ta cần chuyển chữ cái đã được mã hóa sang chữ cái của bảng mã ASCII bằng cách lấy kết quả + 65 (là ký tự A).

+ Ví dụ văn bản đã được mã hóa [i] = W và mã khóa [i] = E ⇔ (22 – 4) % 26 = 18. Tương đương chữ cái S sau đó chuyển sang bảng mã ascii để hiển thị lên máy tính 18 + 65 = 83. Ta thấy 83 là chữ cái S của bảng mã ascii.

CHƯƠNG 3: GIỚI THIỆU CHƯƠNG TRÌNH

3.1 Giới thiệu chương trình

3.1.1 Giao diện chương trình

Hình 3.1 Giao diện chính của chương trình.

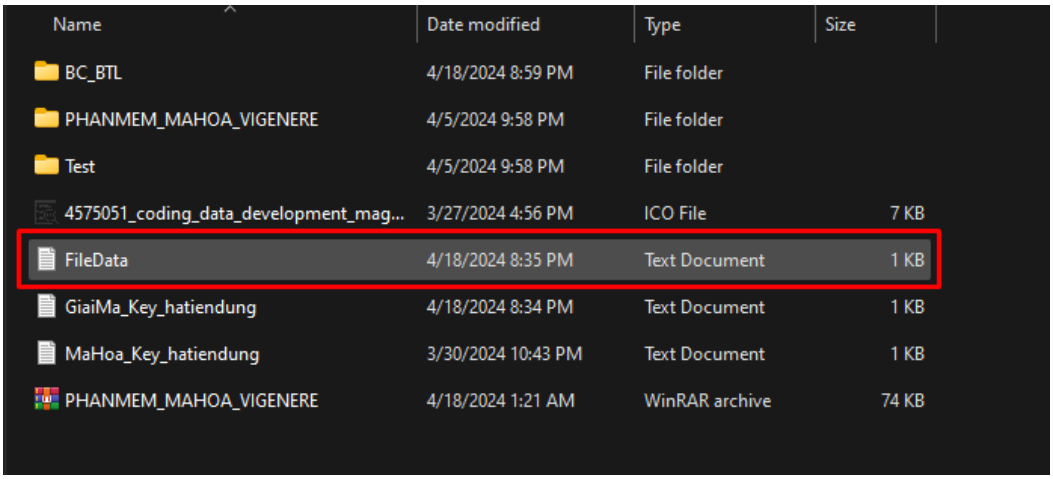
- Bảng thành phần của giao diện chương trình.

Stt	Tên đối tượng	Kiểu item	Ý nghĩa	Ghi chú
1	GIAODIENCHINH	Form	Form chứa các thành phần giao của giao diện.	
2	rtbText	RichTextBox	Khung hiển thị dữ liệu của văn bản cần mã hóa hoặc giải mã.	
3	rtbResult	RichTextBox	Khung hiển thị dữ liệu của văn bản đã được mã hóa hoặc giải mã.	Chỉ đọc
4	txtKey	TextBox	Khung nhập mã khóa để mã hóa hoặc giải mã.	
5	lbFP	Label	Hiển thị đường dẫn tệp văn bản đã được chọn.	
6	btnSLFile	Button	Nút chọn một tệp văn bản cần mã hóa hoặc giải mã.	
7	btnRF	Button	Nút làm mới giao diện của chương trình.	
8	btnEC	Button	Nút chức năng mã hóa đoạn văn bản đã được chọn.	

9	btnDRT	Button	Nút chức năng giải mã đoạn văn bản đã được chọn.	
10	btnSaveFile	Button	Nút chức năng lưu văn bản khi đã được mã hóa hoặc giải mã.	
11	lbDTFile	Label	Chỉ chỗ hiển thị đường dẫn của tệp văn bản.	
12	lbKey	Label	Chỉ chỗ nhập mã khóa để mã hóa hoặc giải mã đoạn văn bản.	
13	lbRS	Label	Chỉ chỗ kết quả của văn bản sau khi được mã hóa hoặc giải mã.	
14	FileVB	OpenFileDialog	Thực hiện chức năng chọn tệp văn bản để mã hóa hoặc giải mã	

3.1.2 Đầu vào và đầu ra

- Đầu vào: Đầu vào của chương trình này là một file .txt được người dùng chọn hoặc một đoạn văn bản được người dùng nhập từ bản phím (Văn bản là các chữ cái tiếng anh không dấu.) và một đoạn mã khóa để tiến hành mã hóa hoặc giải mã.



Hình 3.2 Giao diện tệp hợp lệ để mã hóa

- Đầu ra: Là kết quả của đoạn văn bản sau khi được mã hóa hoặc giải mã với mã khóa mà người dùng đã nhập và đầu ra tiếp theo là xuất một file văn bản .txt nếu người dùng muốn lưu kết quả đó.

3.1.3 Xây dựng chương trình

- Xây dựng hàm kiểm tra văn bản đầu vào có phải là chữ cái hay không.

```
public bool keyIsValid(string Key)
{

    foreach(char c in Key)
    {
        if(!char.IsLetter(c))
        {
            return true;
        }
    }
    return false;
}
```

+ Mô tả : Hàm có tên keyIsValid với tham số là một kiểu string Key. Sử dụng foreach để duyệt toàn bộ các kí tự của tham số được truyền vào. Sử dụng if để kiểm tra với điều kiện !char.IsLetter tức là nếu kí tự đó không phỉa là chữ cái thì trả về true. Sau khi chạy hết vòng lặp mà không có kí tự nào khác chữ cái thì trả về false.

- Xây dựng hàm mã hóa văn bản.

```
public void enCrypt(ref StringBuilder data, string key)
{
    int tmpKey, tmpData;
    for(int i = 0; i < data.Length; i++)
    {
        data[i] = char.ToUpper(data[i]);
    }
    key = key.ToUpper();
    int j = 0;
    for(int i = 0; i < data.Length; i++)
    {
        if(Char.IsLetter(data[i]))
        {
            tmpKey = key[j] - 65;
            tmpData = data[i] - 65;
            tmpData = (tmpData + tmpKey) % 26;
            data[i] = (char)(tmpData + 'A');
        }
        j = (j + 1 == key.Length) ? 0 : j + 1;
    }
}
```

+ Mô tả: Hàm có tên là enCrypt với 2 tham số là đối tượng StringBuilder data và string key. Khai báo 3 biến int tmpKey = lưu giá trị số của khóa, tmpData lưu giá trị số của dữ liệu, j lưu giá trị vị trí của mảng mã khóa. Tiếp theo sử dụng for để chuyển tất cả chữ cái của data thành chữ in hoa `data[i] = char.ToUpper(data[i]);` Và sau đó chuyển các chữ cái của mã khóa thành chữ hoa `key = key.ToUpper();` . Tiếp theo sử dụng for để tiến hành mã hóa đoạn văn bản với i = 0 , i < độ dài của dữ liệu, i tự tăng sau mỗi vòng lặp. Khi điều kiện chữ cái thứ i của dữ liệu là chữ cái ta gán tmpKey = Key thứ j – 65 và tmpData = data thứ i – 65 để chuyển số từ bảng mã ascii thành số theo thứ tự bảng chữ cái từ 0 – 25. Tiếp theo gán tmpData = (tmpData + tmpKey) chia lấy dư cho 26 ta được kết quả của chữ cái được mã hóa với khóa. Sau đó gán data thứ i = tmpData + ‘A’ tức là data thứ i sẽ = kết quả mã hóa cộng

với 65 để chuyển kí tự số đó thành kí tự trong bảng mã ascii. Nếu khi j + 1 = với độ dài của mã khóa j sẽ được gán lại = 0 và ngược lại j = j + 1.

- Xây dựng hàm giải mã văn bản.

```
public void decCrypt(ref StringBuilder data, string key)
        {
            int tmpKey, tmpData;
            for (int i = 0; i < data.Length; i++)
            {
                data[i] = Char.ToUpper(data[i]);
            }
            key = key.ToUpper();
            int j = 0;
            for(int i = 0; i < data.Length; i++)
            {
                if(Char.IsLetter(data[i]))
                {
                    tmpKey = key[j] - 65;
                    tmpData = data[i] - 65;
                    tmpData = ((tmpData - tmpKey) + 26) % 26;
                    data[i] = (char)(tmpData + 'A');
                }
                j = (j + 1 == key.Length) ? 0 : j + 1;
            }
        }
```

+ Mô tả: Hàm có tên là decCrypt với 2 tham số là đối tượng StringBuilder data và string key. Khai báo 3 biến int tmpKey = lưu giá trị số của khóa, tmpData lưu giá trị số của dữ liệu, j lưu giá trị vị trí của mảng mã khóa. Tiếp theo sử dụng for để chuyển tất cả chữ cái của data thành chữ in hoa `data[i] = char.ToUpper(data[i]);` Và sau đó chuyển các chữ cái của mã khóa thành chữ hoa `key = key.ToUpper();` . Tiếp theo sử dụng for để tiến hành giải mã đoạn văn bản với i = 0 , i < độ dài của dữ liệu, i tự tăng sau mỗi vòng lặp. Khi điều kiện chữ cái thứ i của dữ liệu là chữ cái ta gán tmpKey = Key thứ j – 65 và tmpData = data thứ i – 65 để chuyển số từ bảng mã ascii thành số theo thứ tự bảng chữ cái từ 0 – 25. Tiếp theo gán tmpData = (tmpData - tmpKey) + 26 chia lấy dư cho 26 ta được kết quả của chữ cái được giải mã với khóa. Sau đó gán data thứ i = tmpData + ‘A’ tức là data thứ i sẽ = kết quả giải mã cộng với 65 để chuyển kí tự số đó thành kí tự trong bảng mã ascii. Nếu khi j + 1 = với độ dài của mã khóa j sẽ được gán lại = 0 và ngược lại j = j + 1.

- Xây dựng hàm đọc file văn bản người dùng đã chọn.

```
        public string readDataFile(string path)
        {
            string data = "";
            using (StreamReader reader = new StreamReader(path))
            {
                 data = reader.ReadToEnd();
            }
            return data;
        }
```

+ Mô tả: Hàm có tên readDataFile với tham số là một string path. Khai báo một biến string data = “”. Tiếp theo sử dụng một lớp StreamReader có tên reader = một lớp

StreamReader được truyền tham số path vào. Ở trong using gán data = reader.ReadToEnd(); Tức là đọc từ đầu đến cuối của file văn bản đó. Sau đó trả về data đó.

- Thiết lập sự kiện cho nút btnSLFile (Khi người dùng chọn nút).

```
        private void btnSLFile_Click(object sender, EventArgs e)
        {
            try
            {
                FileVB.ShowDialog();
                string filePath = FileVB.FileName;
                string fileExt = Path.GetExtension(FileVB.FileName);
                if(fileExt != ".txt")
                {
                    MessageBox.Show("File bạn chọn không phải là một file văn bản .txt!",
"Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
                else
                {
                    lbFP.Text = filePath.Trim();
                    rtbText.Text = fts.readDataFile(filePath).Trim();
                    dataText = fts.readDataFile(filePath).Trim();
                    strBData = new StringBuilder(dataText);
                }
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message, "ERROR!!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
```

+ Mô tả: Gọi đối tượng FileVB.ShowDialog(); để hiển thị cửa sổ chọn file. Gán biến string file Path = FileVB.FileName (đường dẫn của văn bản). Gán biến string fileExt = Path.GetExtension(FileVB.FileName) để lấy đuôi định dạng của file. Tiếp theo kiểm tra file có phải là file hợp lệ không, nếu không sẽ hiển thị thông báo. Sau đó nếu file đã hợp lệ gán đối tượng lbFP.Text để hiển thị đường dẫn đã được chọn , gán đối tượng rtb.Text và biến string dataText = fts.readDataFile(filePath) là hàm đọc file văn bản đã được xây dựng trước đó để hiển thị văn bản lên màn hình và lưu dữ liệu văn bản vào biến dataText. Cuối cùng gắn lớp StringBuilder strBData = StringBuilder với tham số là dataText.

- Thiết lập sự kiện cho nút btnEC (Khi người dùng chọn nút).

```
        private void btnEC_Click(object sender, EventArgs e)
        {
            try
            {
                if(rtbText.Text == "" || rtbText.Text == null)
                {
                    MessageBox.Show("Empty data file!", "Warning!", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
                else if (txtKey.Text == null || txtKey.Text == "")
                {
                    MessageBox.Show("Empty key code!", "Warning!", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
                else if(fts.keyIsValid(txtKey.Text.Trim()))
                {
                    MessageBox.Show("The key has invalid characters!", "Warning!",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
                else
                {
                    fts.enCrypt(ref strBData, txtKey.Text.Trim());
                    rtbResult.Text = Convert.ToString(strBData);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "ERROR!!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
```

+ Mô tả: Kiểm tra nếu rtbText rỗng hoặc null thì thông báo ra màn hình. Kiểm tra mã khóa nếu rỗng hoặc null thì thông báo ra màn hình. Kiểm tra mã khóa không hợp lệ nếu không hợp lệ thông báo ra màn hình. Cuối cùng khi tất cả điều kiện được thỏa mãn ta gọi hàm mã hóa enCrypt đã được xây dựng trước đó và truyền vào StringBuilder strData, txtKey.Text); Sau đó gán cho đối tượng rtbResult.Text = strBData để hiển thị kết quả mã hóa lên màn hình.

- Thiết lập sự kiện cho nút btnDRT (Khi người dùng chọn nút).

```
        private void btnDRT_Click(object sender, EventArgs e)
        {
            try
            {
                if (rtbText.Text == "" || rtbText.Text == null)
                {
                    MessageBox.Show("Empty data file!", "Warning!", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
                else if (txtKey.Text == null || txtKey.Text == "")
                {
                    MessageBox.Show("Empty key code!", "Warning!", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
                else if (fts.keyIsValid(txtKey.Text.Trim()))
                {
                    MessageBox.Show("The key has invalid characters!", "Warning!",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
```

```
                else
                {
                    fts.decCrypt(ref strBData, txtKey.Text.Trim());
                    rtbResult.Text = Convert.ToString(strBData);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "ERROR!!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
```

+ Mô tả: Kiểm tra nếu rtbText rỗng hoặc null thì thông báo ra màn hình. Kiểm tra mã khóa nếu rỗng hoặc null thì thông báo ra màn hình. Kiểm tra mã khóa không hợp lệ nếu không hợp lệ thông báo ra màn hình. Cuối cùng khi tất cả điều kiện được thỏa mãn ta gọi hàm mã hóa decCrypt đã được xây dựng trước đó và truyền vào StringBuilder strData, txtKey.Text); Sau đó gán cho đối tượng rtbResult.Text = strBData để hiển thị kết quả giải mã lên màn hình.

- Thiết lập sự kiện cho nút btnSaveFile (Khi người dùng chọn nút).

```
private void btnSaveFile_Click(object sender, EventArgs e)
        {
            using (SaveFileDialog saveDialog = new SaveFileDialog())
            {
                saveDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*"; //
Chỉ định loại file
                saveDialog.Title = "Chọn nơi lưu file"; // Tiêu đề của hộp thoại

                if (saveDialog.ShowDialog() == DialogResult.OK)
                {
                    try
                    {
                        string fileDerectory =
Path.GetDirectoryName(saveDialog.FileName);
                        string fileName =
Path.GetFileNameWithoutExtension(saveDialog.FileName);
                        string fileExtension = Path.GetExtension(saveDialog.FileName);
                        saveDialog.FileName = Path.Combine(fileDerectory,
$"{fileName}_Key_{txtKey.Text}{fileExtension}");
                        // Ghi đoạn string vào file được chọn
                        File.WriteAllText(saveDialog.FileName, rtbResult.Text.Trim());
                        MessageBox.Show("Lưu file thành công!");
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message, "ERROR!!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                }
            }
        }
```

+ Mô tả: Sử dụng lớp SaveFileDialog saveDiaLog. Trong using saveDiaLog.Filter = “Textfile (*.txt)|*.txt|All files (*.*)|*.*” để chỉ định loại file được lưu có đuôi .txt. Tiếp theo để hiển thị tiêu đề ta sử dụng saveDiaLog.Title. Sau đó nếu khi saveDiaLog.ShowDiaLog() == DiaLogReSult.OK tức là khi người dùng bấm OK. Ta gán biến fileDerectory = Path.GetDirectory(saveDiaLog.FileName) để lấy đường dẫn đến file. Gán fileName = Path.GetFileNameWithoutExtension(saveDiaLog.FileName) để lấy mỗi tên file người dùng

đã nhập, gán fileExtension = Path.GetExtension(saveDiaLog.FileName) để lấy đuôi định dạng file. Sau đó gán cho saveDiaLog.FileName để định dạng lại tên file. = Path.Combie(fileDerectory, $"{fileName}_Key_{txtKey.Text}{fileExtension}"); Để thêm mã khóa vào sau tên file của người dùng đã đặt. Sau đó tiến hành lưu file vào thư mục đã định bằng cách gọi lớp File.WriteAllText và truyền hai tham số saveDiaLog là đường dẫn của file và rtbResult.Text là dữ liệu của file. Cuối cùng thông báo lưu file thành công cho người dùng.

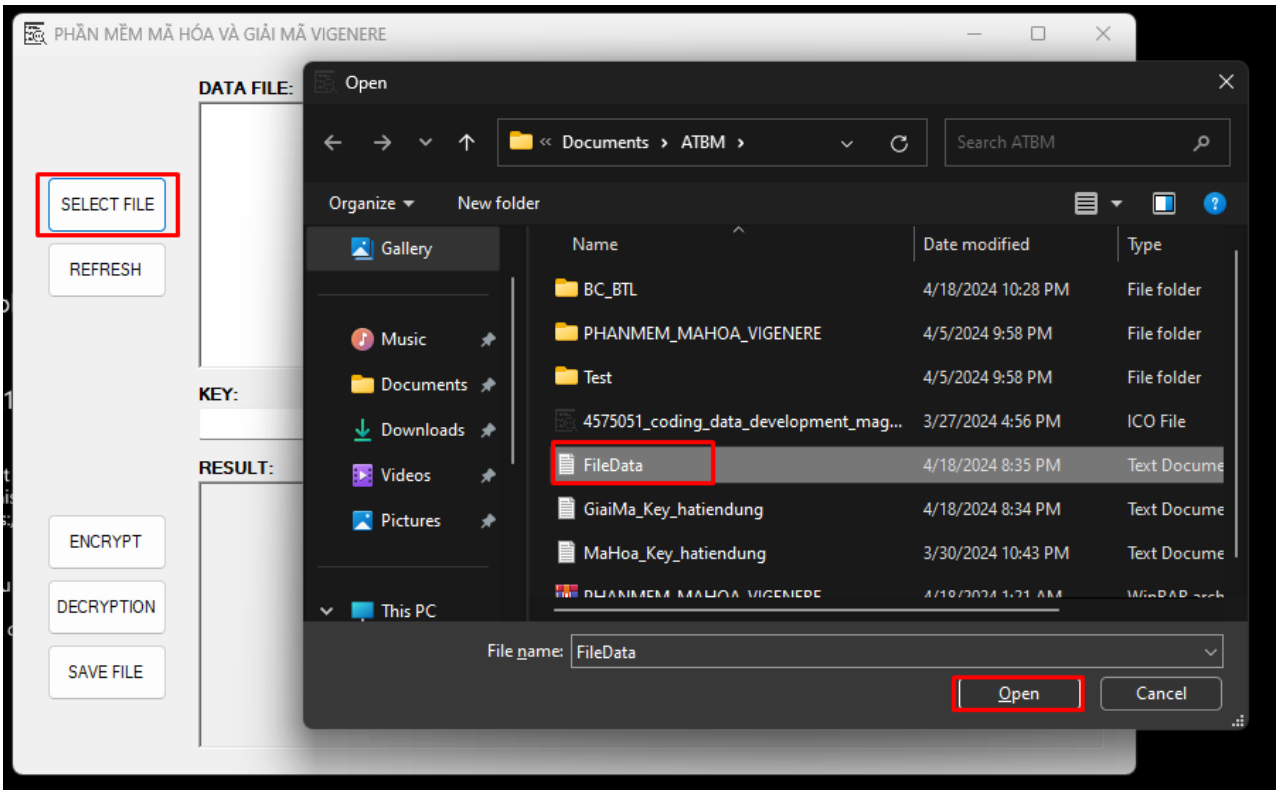
- Thiết lập sự kiện cho ô văn bản rtbText (Khi đoạn văn bản được thay đổi).

```
        private void rtbText_TextChanged(object sender, EventArgs e)
        {
            strBData = new StringBuilder(rtbText.Text.Trim());
        }
```

+ Mô tả: Khi dữ liệu của rtbText bị thay đổi ta sẽ gắn StringBuilder strBData = lớp StringBuilder có tham số của rtbText.Text mới nhất.

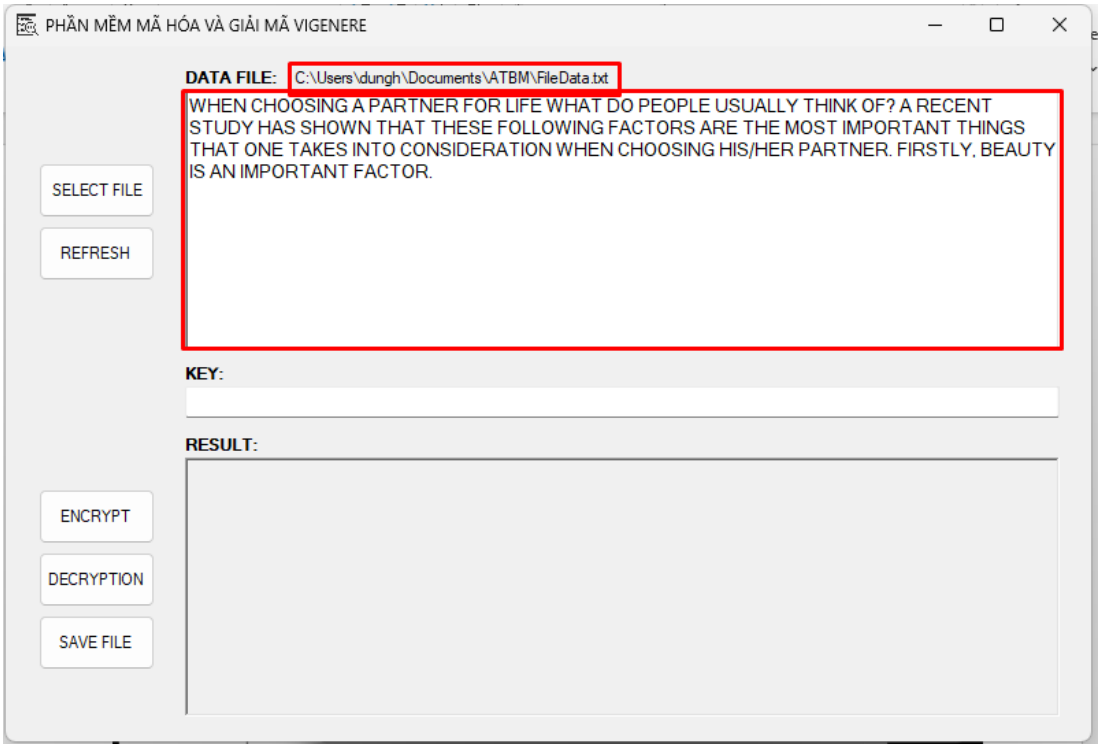
3.2 Kết quả thu được

- Để mã hóa văn bản ở giao diện chính chọn select file. Hệ thống sẽ hiển thị giao diện chọn file, người dùng chọn file muốn mã hóa và ấn open.



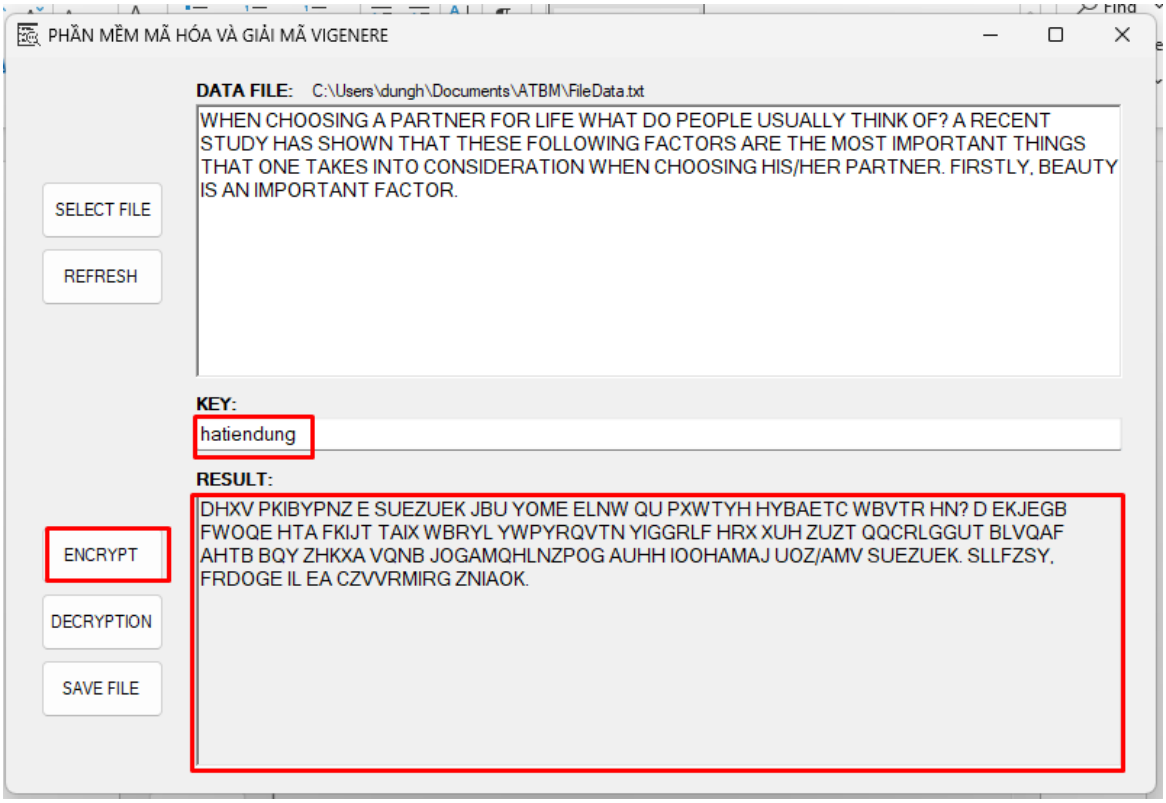
Hình 3.3 Tiến hành chọn file để mã hóa

- Kết quả khi người dùng chọn file hợp lệ.



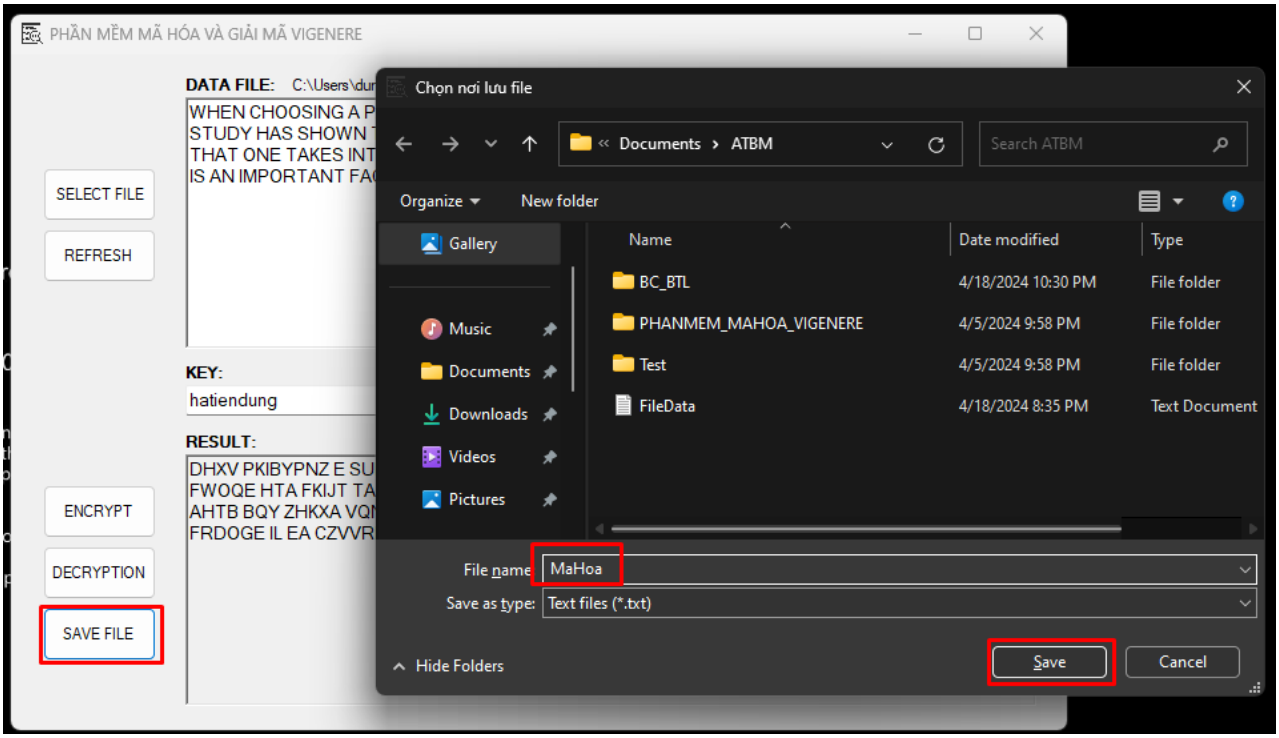
Hình 3.4 Kết quả sau khi chọn file

- Để tiến hành mã hóa người dùng nhập mã khóa và chọn nút encrypt. Kết quả mã hóa sẽ được hiển thị dưới khung result.



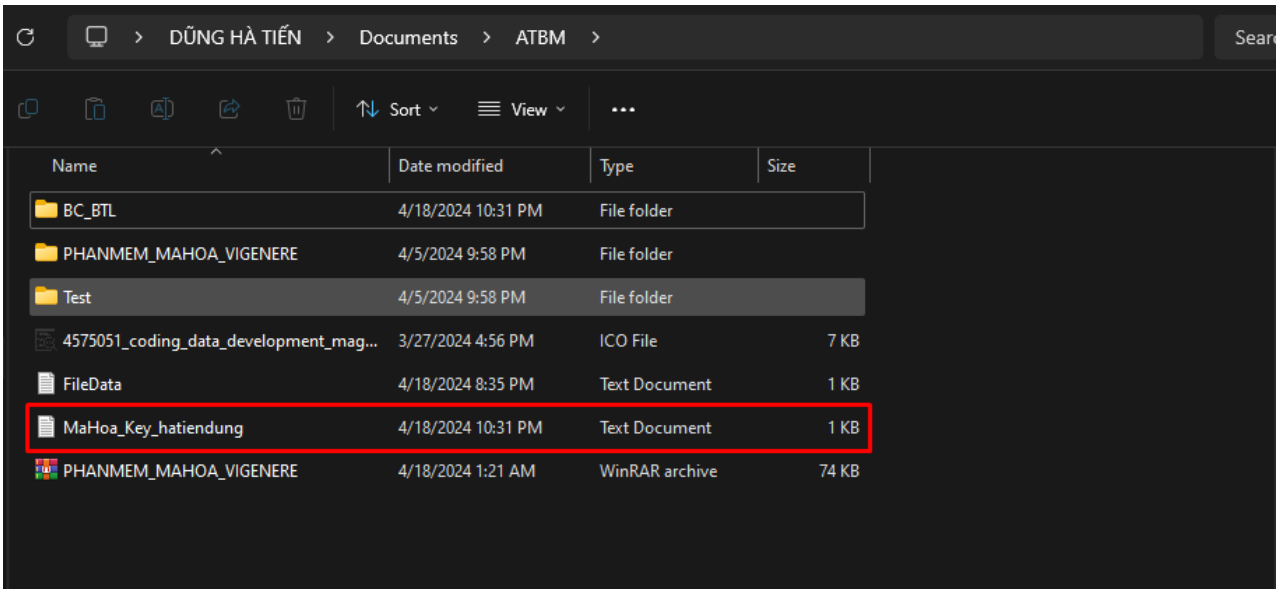
Hình 3.5 kết quả khi nhập mã khóa và ấn nút encrypt để mã hóa

- Để lưu file sau khi mã hóa người dùng chọn nút save file. Sau đó tiến hành nhập tên file và ấn save.



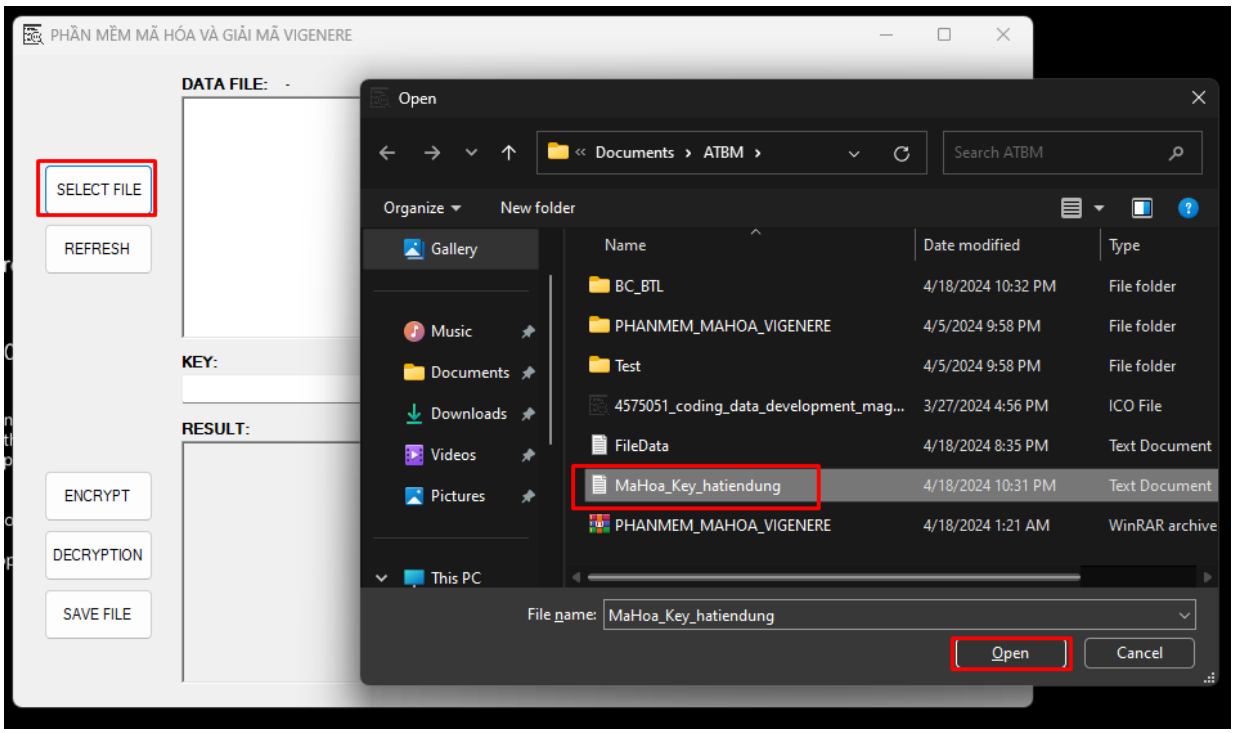
Hình 3.6 Tiến hành lưu file kết quả mã hóa.

- Kết quả sau khi lưu file.



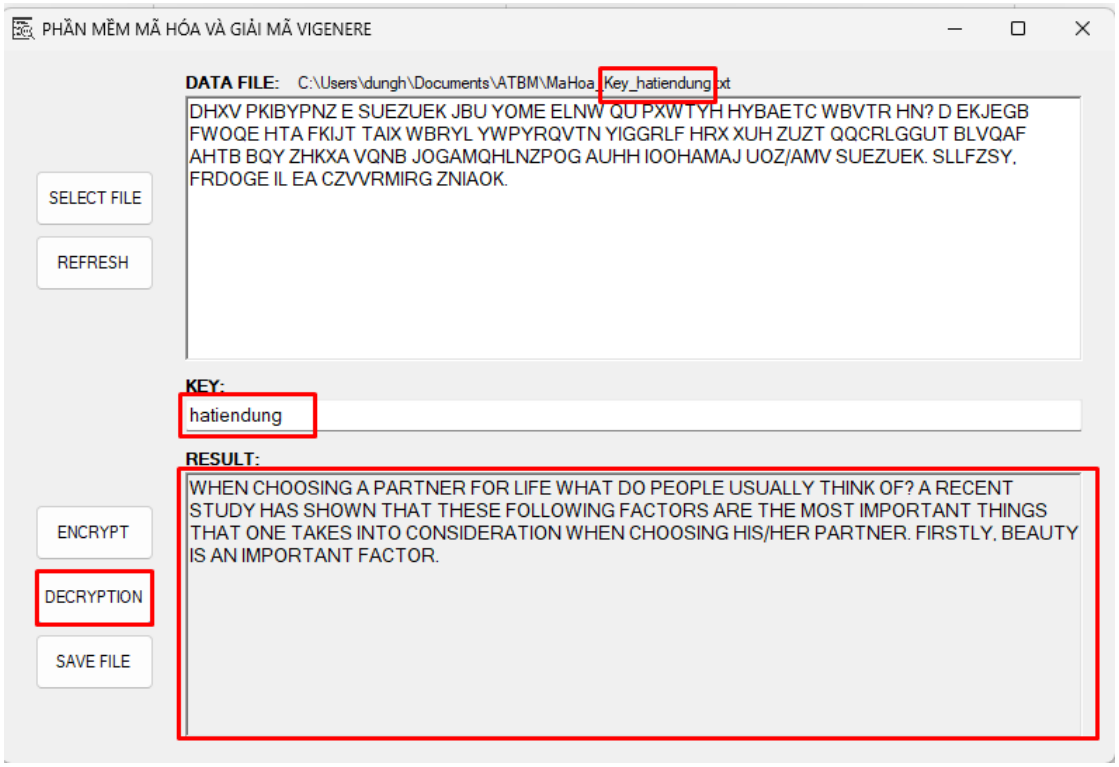
Hình 3.7 Kết quả sau khi lưu file.

- Để tiến hành giải mã người dùng chọn nút select file sau đó chọn file văn bản cần giải mã và ấn open.



Hình 3.8 Tiến hành chọn file để giải mã.

- Sau đó người dùng nhập mã khóa có trong tên file và ấn nút decryption. Kết quả giải mã sẽ được hiển thị dưới khung result.



Hình 3.9 Kết quả giải mã sau khi nhập mã khóa và ấn nút decrytion.

KẾT LUẬN

Kết luận

- Qua quá trình học tập, nghe giảng sau đó bắt tay vào thực hiện xây dựng chương trình mã hóa và giải mã văn bản với hệ mã hóa vigenere chúng em cũng đã hoàn thành được một chương trình cơ bản đáp ứng được yêu cầu đã đặt ra.

Hạn chế và hướng phát triển của đề tài

- Vì thời gian và kiến thức có hạn nên chương trình có lẽ vẫn còn nhiều hạn chế. Chương trình chỉ mang tính học tập áp dụng vào thực tế chưa cao.

- Chưa tối ưu code.

- Từ nhưng kết quả và hạn chế trên trong tương lai chúng em sẽ nghiên cứu và hoàn thiện, bám sát với nhu cầu thực tế hơn, tạo ra một ứng dụng bắt mắt dễ dùng, mã hóa và giải mã được cái file văn bản khác ngoài văn bản định dạng .txt và từ đó có thể phát triển các ứng dụng chat qua mạng,… có sử dụng hệ mã hóa vigenere để nâng cao kiến thức cũng như hiểu biết của mình hơn.

TÀI LIỆU THAM KHẢO

1. Một số tài liệu do giảng viên cung cấp.
2. Một số kiến thức lý thuyết tham khảo công cụ AI: Chat GPT, Bard, Bing AI.
3. Giáo trình An toàn và bảo mật thông tin: Phần 2 - PGS.TS. Đàm Gia Mạnh, TS. Nguyễn Thị Hội (Chủ biên) - TaiLieu.VN
4. Mật mã Vigenère – Wikipedia tiếng Việt