TRƯỜNG ĐẠI HỌC LÂM NGHIỆP

KHOA HỆ THỐNG THÔNG TIN

========&&&========



**BÀI THI CUỐI KỲ**

**MÔN : AN TOÀN VÀ BẢO MẬT THÔNG TIN**

**CHỦ ĐỀ : MÃ HÓA FILE BẰNG PHƯƠNG THỨC AES**

**Giáo viên hướng dẫn : Mai Hà An**

**Nhóm thực hiện : Nguyễn Trắc Thanh Tùng**

**Nguyễn Công Văn**

MỤC LỤC

[**LỜI MỞ ĐẦU** 3](#_Toc137563106)

[**GIỚI THIỆU ĐỒ ÁN** 4](#_Toc137563107)

[**LỊCH TRÌNH HOẠT ĐỘNG NHÓM** 4](#_Toc137563108)

[PHƯƠNG THỨC MÃ HÓA AES 5](#_Toc137563109)

[**1.1**Mã hóa AES  là gì? 5](#_Toc137563110)

[1.2 AES hoạt động như thế nào? 5](#_Toc137563111)

[1.3 Các tiêu chí cần có ở AES 6](#_Toc137563112)

[1.4 Sự khác biệt giữa AES-128 và AES-256 7](#_Toc137563113)

[1.5 Sự khác biệt giữa AES và RSA 7](#_Toc137563114)

[1.6 Phân biệt AES và DES 7](#_Toc137563115)

[1.7 Tấn công mã hóa AES 8](#_Toc137563116)

[1.8 AES có an toàn không? 8](#_Toc137563117)

[**CÂU LỆNH VÀ GIẢI THUẬT** 9](#_Toc137563118)

[**TỔNG KẾT** 22](#_Toc137563119)

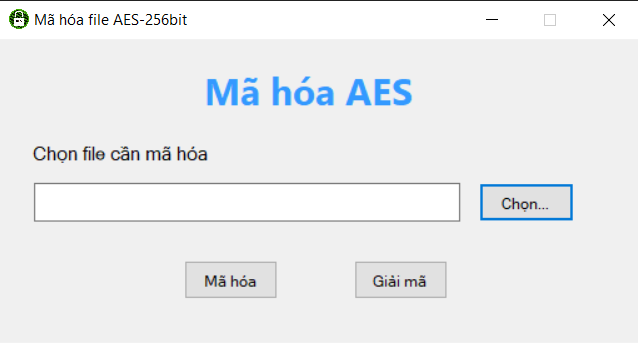
**LỜI MỞ ĐẦU**

Trong những năm gần đây, dưới sự phát triển mạnh mẽ của thời đại công nghệ số 4.0, khiến cho công việc của mọi người trở nên thuận tiện và dễ dàng hơn nhờ những công nghệ mới, những phần mềm được phát triển hỗ trợ cho các công việc đặc thù. Việc trao đổi thông tin qua internet hay lưu trữ trên phần cứng, đám mây… ngày càng phổ biến nhưng cái mà đa số người dùng quan tâm chỉ là những thông tin của họ được lưu trữ và những họ có thể trả đổi chúng trực tuyến mà không hề biết rằng những thông tin đó hoàn toàn có thể bị đánh cắp bởi những tin tặc. Trên thực tế, đã có rất nhiều vụ đánh cắp dữ liệu lơn trên thế giới ví dụ như equifax(2017) khiến cho thông tin cá nhân của 143 triệu người dùng (gồm ngày sinh, địa chỉ, số an sinh xã hội, số bằng lái xe) đã bị vi phạm. Đồng thời, 209.000 khách hàng đã bị lộ thông tin thẻ tín dụng, hay vụ lọt dữ liệu nghiêm trọng nhất đầu thế kỷ 21 phải kể đến Yahoo!. Năm 2014, Yahoo! đã tiết lộ rằng họ đã phải chịu tấn công vi phạm dữ liệu, làm ảnh hưởng đến 500 triệu tài khoản. Không chỉ bị hack, các tài khoản này còn bị hacker sử dụng để tiếp tục lừa đảo người thân, bạn bè của chính chủ, còn rất nhiều vụ tin tặc tấn công khác trên thế giới. Chính vì thế mà việc bảo mật dữ liệu đã trở thành một khía cạnh rất quan trọng trong thời điểm mà công nghệ kĩ thuật số phát triển mạnh mẽ như hiện nay. Có thể là thông tin cá nhân của họ trên các trang mạng xã hội hoặc các nền tảng online khác hoặc dữ liệu offline trên máy tính và thiết bị di động của họ, tất cả có khả năng bị đánh cắp bởi các tin tặc. Vì vậy, điều quan trọng là phải bảo vệ dữ liệu của bạn bằng mọi cách có thể. Mặc dù rất khó để tự bảo vệ mình trên các nền tảng online, nhưng bạn phải cẩn thận đối với thông tin mà bạn tải lên hoặc chia sẻ, nhưng đối với những thông tin bạn đang lưu trữ trên thiết bị cá nhân bạn hoàn toàn có thể bảo mật chúng 1 cách an toàn. Các file thông tin cá nhân, công việc… hoàn toàn có thể được bảo mật bằng các ứng dụng hỗ trợ cho việc mã hóa dữ liệu nhằm đảm bảo dữ liệu được an toàn. Việc mã hóa dữ liệu ngăn không cho những người dùng khác có quyền truy cập vào cùng một máy tính xem dữ liệu đó. **Dù bạn lưu trữ dữ liệu dưới bất kỳ hình thức nào, trên USB hoặc trực tiếp trên ô cứng máy tính bạn cũng cần phải thiết lập thêm 1 hay nhiều lớp bảo vệ cho tập tin hoặc thư mục để bảo đảm an toàn.** Và việc thiết lập mật khẩu cho tập tin hoặc thư mục là cách an toàn nhất cho việc mã hóa và bảo vệ dữ liệu. Tập tin hoặc thư mục của bạn sẽ được mã hóa và chỉ có thể mở hoặc sử dụng bằng cách khai báo mật khẩu. Dưới đây chúng tôi sẽ giới thiệu cho bạn 1 phần mềm có thể hỗ trợ cho việc bảo mật dữ liệu của bạn một cách an toàn, với 1 giao diện đơn giản thân thiện với người dùng, và sử dụng thuật toán **Advanced Encryption Standard(AES)** hay còn gọi là tiêu chuẩn mã hóa nâng cao theo phương thức mật mã khối. được chính phủ Hoa Kỳ tin dùng để bảo vệ dữ liệu,

**GIỚI THIỆU ĐỒ ÁN**

**Đề tài:THIẾT KẾ GIAO DIỆN MÃ HÓA FILE BẰNG PHƯƠNG THỨC AES**

**Ngôn ngữ lập trình : C#**



**LỊCH TRÌNH HOẠT ĐỘNG NHÓM**

|  |  |  |  |
| --- | --- | --- | --- |
| Tên thành viên | Thời gian làm việc | | |
| 18/4/2023 | 06/5/2023 | 10/6/2023 |
| Nguyễn Công Văn | Tìm hiểu các nội dung liên quan | Lên ý tưởng thiết kế giao diện  Chỉnh sửa, bổ sung code giao diện | Làm báo cáo Work,  PowerPoint |
| Nguyễn Trắc Thanh Tùng | Tìm hiểu các nội dung liên quan | Code giao diện  Code mã hóa AES-256bit | Đóng góp ý kiến, bổ sung và chỉnh sửa Work |

# PHƯƠNG THỨC MÃ HÓA AES

## **1.1**Mã hóa AES  là gì?

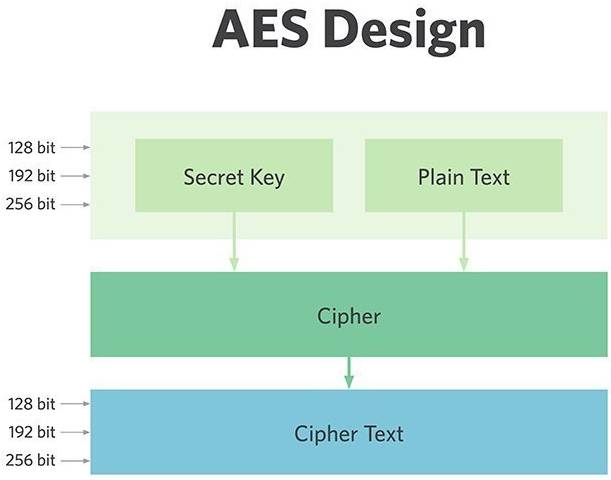
**Advanced Encryption Standard (AES) hay còn được gọi là tiêu chuẩn mã hóa nâng cao theo phương pháp mật mã khối**. Với ưu thế bảo mật cao nó đã được chính phủ Hoa Kỳ lựa chọn để bảo vệ dữ liệu, thông tin cho các tổ chức, doanh nghiệp mà người dùng.

Các tác vụ của AES được thực hiện ở cả phần cứng và phần mềm trên nhiều thiết bị để mã hóa dữ liệu nhạy cảm. Sự có mặt của nó đã góp phần bảo đảm an toàn cho máy tính của chính phủ, an ninh mạng và tạo một rào chắn vững chắc để bảo vệ dữ liệu.

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| độ bảo mật cao, khả năng mã hóa và giải mã nhanh chóng và hiệu quả, khả năng sử dụng với các khối dữ liệu có kích thước khác nhau và được sử dụng rộng rãi trong các ứng dụng bảo mật thông tin | Chi phí tính toán cao hơn với các thuật toán khác |

## 1.2 AES hoạt động như thế nào?

AES gồm ba mật mã khối AES-128, AES-192, AES-256 tương ứng với độ dài của key là 128 bit, 192 bit và 256 bit. Số vòng của key khác nhau, cụ thể 10 vòng cho 128 bit, 12 vòng cho 192 bit và 14 vòng cho 256 bit. Mỗi vòng đều thực hiện ba bước thay thế, biến đổi và hòa trộn khối plain text (văn bản thuần túy) đầu vào để biến nó thành Ciphertext (văn bản đã mã hóa).



Phân loại theo ba cấp độ: bảo mật, bí mật, tối mật. Tất cả các độ dài của key từ 128, 192 và 256 bit đều được dùng ở cấp độ bảo mật, bí mật. Riêng với những thông tin tối mật để đảm bảo không xảy ra bất cứ sai sót nào phải cần đến key 192 hoặc 256 bit. Mật mã sẽ dùng một key riêng tư để mã hóa và giải mã dữ liệu và tất nhiên cả người gửi và người nhận đều phải nhận biết và sử dụng được key này.

## 1.3 Các tiêu chí cần có ở AES

NIST đưa ra yêu cầu đối với AES đó là phải sử dụng phương pháp mã hóa khối với độ dài của key là 128, 192 và 256 bit để mã hóa và giải mã dữ liệu. Ngoài ra AES phải đáp ứng được những tiêu chí sau:

* **Bảo vệ**: Đây là một trong những tính năng hàng đầu AES cần phải có để đánh bại các đối thủ khác. Nó phải có khả năng chống lại các cuộc tấn công mạnh, quy mô lớn.
* **Chi phí**: AES mở ra nhiều cơ hội cho người dùng bằng cách phát hành trên toàn cầu và miễn phí bản quyền.
* **Khả năng thực hiện**: Linh hoạt, phù hợp và đơn giản chính là 3 yếu tố quan trọng hội tụ ở AES để đáp ứng trọn vẹn nhu cầu của người dùng.

## 1.4 Sự khác biệt giữa AES-128 và AES-256

mã hóa 256 bit khó đoán hơn rất nhiều so với key 128 bit. Vì lẽ đó mà nó rất mạnh mẽ và bảo vệ dữ liệu trước các cuộc tấn công brute – force. Trong tương lai thì chưa thể biết được nhưng hiện tại việc phá vỡ thuật toán này là điều không tưởng. Và có thể xem AES-256 là giải pháp hoàn hảo để các tổ chức bảo mật dữ liệu của mình.

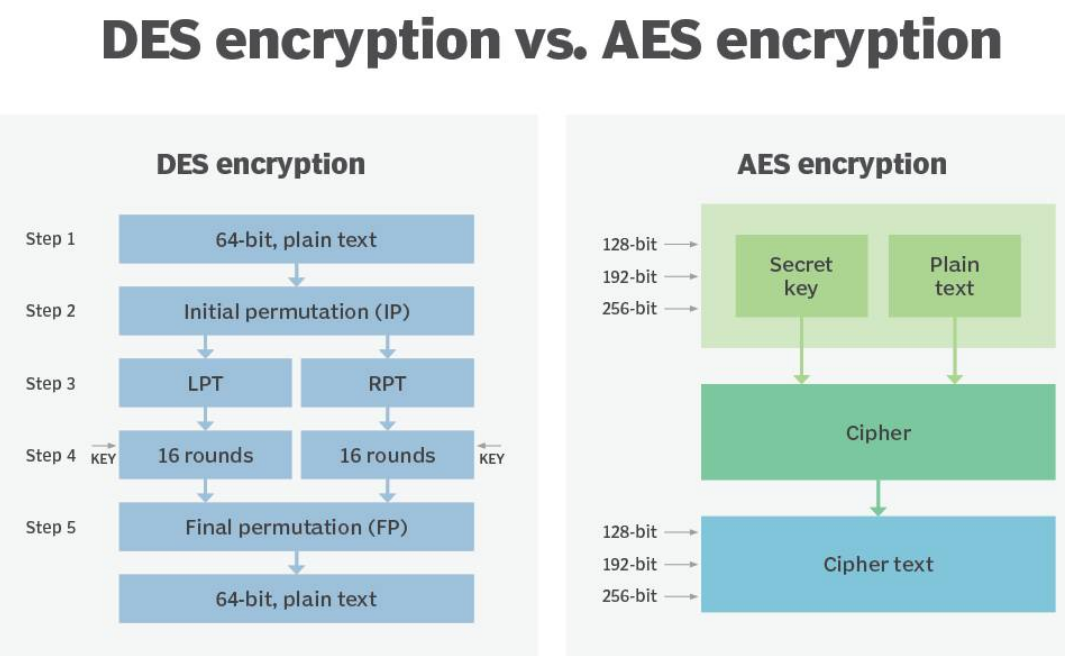
Tuy nhiên một điểm hạn chế của key 256bit đó là nó đòi hỏi sức mạnh xử lý cao hơn key 128 bit và chắc chắn rằng thời gian để thực thi cũng lâu hơn. Do đó đối với các thiết bị nhỏ hoặc với nguồn điện tải trọng thấp thì nên cân nhắc sử dụng key 128 bit.

## 1.5 Sự khác biệt giữa AES và RSA

AES được cá nhân, tổ chức sử dụng để bảo vệ những dữ liệu ở trạng thái nghỉ (dữ liệu lưu trữ trong kho). Ứng dụng của AES có thể kể đến như: tự mã hóa ổ đĩa, mã hóa lưu trữ, mã hóa cơ sở dữ liệu. Trong khi đó RSA lại được ứng dụng để kết nối các website trong các trình duyệt và nhiều lĩnh vực khác.

Một điểm khác biệt nữa đó là trong khi AES sử dụng key riêng tư thì RSA lại dùng key công khai. Bên cạnh đó hiệu suất hoạt động của RSA chậm hơn so với AES. Vì thế giải pháp tốt nhất để bảo vệ việc truyền dữ liệu từ ở khoảng cách xa đó là kết hợp cả mã hóa RSA với AES.

## 1.6 Phân biệt AES và DES



Cả AES và DES đều là mật mã khối đối xứng nhưng AES mang đến hiệu quả cao bởi độ dài key mà nó sở hữu. Đem lên bàn cân có thể thấy các key 128 bit, 192 bit và 256 bit của AES mạnh gấp nhiều lần so với 56 bit của DES. Bên cạnh đó mã hóa AES cũng nhanh hơn DES.

## 1.7 Tấn công mã hóa AES

Mặc dù mã hóa AES là thuật toán mạnh mẽ nhưng không tránh khỏi các cuộc tấn công. Nhiều nhà nghiên cứu đã công bố danh sách các cuộc tấn công, chẳng hạn như:

* Năm 2009 cryptanalysis (công cụ phân tích mật mã) đã tìm cách bẻ key của hệ thống AES.
* Năm 2009, AES-128 đã trở thành mục tiêu của một cuộc tấn công, tuy nhiên mối đe dọa này không gây thiệt hại gì.
* Đáng quan ngại nhất với mã hóa AES là phải đối mặt với side-channel attack (tấn công kênh phụ). Để xác định được key của mã hóa AES các cuộc tấn công này đã sử dụng những thông tin về thời gian, rò rỉ điện từ, manh mối âm thanh…

## 1.8 AES có an toàn không?

AES nếu được triển khai đúng quy trình thì sẽ đảm bảo an toàn tuyệt đối. Thế nhưng một điều cần lưu ý đó là bất kỳ một hệ thống nào cũng có thể bị tấn công nếu hacker biết được key mã hóa. Do đó các key mã hóa AES phải được bảo vệ bằng nhiều cách khác nhau như dùng mật khẩu mạnh, xác thực, tường lửa hay phần mềm chống độc hại. Ngoài ra các tổ chức phải đào tạo nhân viên để nâng cao nhận thức của họ trong việc bảo mật thông tin, tránh bị kẻ xấu lợi dụng.

**CÂU LỆNH VÀ GIẢI THUẬT**

**FORM passdecryption**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace atbmtt

{

public partial class passdecryption : Form

{

public passdecryption()

{

InitializeComponent();

}

private Form1 frm; // tạo biến để lưu formmain

public passdecryption(Form1 fm) // tạo tham số kiểu formmain cho hàm tạo

{

InitializeComponent();

frm = fm; // gán formmain được truyền vào cho biến fm

}

private void passdecryption\_Load(object sender, EventArgs e)

{

txtpass.PasswordChar = '\*';

}

public delegate void TruyenPasswordMH(string txt);

public TruyenPasswordMH TruyenPass;

private void button1\_Click(object sender, EventArgs e)

{

string passwordMH = txtpass.Text; // Lấy pass từ textbox

// Kiểm tra xem password có rỗng không

if (passwordMH.Length > 0)

{

if (TruyenPass != null)

{

TruyenPass(passwordMH);

}

this.Close();

}

else

{

// Hiển thị thông báo lỗi

MessageBox.Show("Hãy nhập mật khẩu!", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

// Kiểm tra trạng thái của checkbox

if (checkBox1.Checked)

{

// Nếu checkbox được tích, đặt thuộc tính PasswordChar của textbox là '\0' để hiển thị các ký tự

txtpass.PasswordChar = '\0';

}

else

{

// Nếu checkbox không được tích, đặt thuộc tính PasswordChar của textbox là '\*' để ẩn các ký tự

txtpass.PasswordChar = '\*';

}

}

}

}

**FORM Pass**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Net.Mime.MediaTypeNames;

namespace atbmtt

{

public partial class Pass : Form

{

public Pass()

{

InitializeComponent();

}

private Form1 fm; // tạo biến để lưu formmain

public Pass(Form1 f) // tạo tham số kiểu formmain cho hàm tạo

{

InitializeComponent();

fm = f; // gán formmain được truyền vào cho biến fm

}

private void pass\_Load(object sender, EventArgs e)

{

PassWork.PasswordChar = '\*';

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

public delegate void TruyenPassword(string txt);

public TruyenPassword truyen;

private void button1\_Click(object sender, EventArgs e)

{

string password = PassWork.Text; // Lấy pass từ textbox

// Kiểm tra xem password có rỗng không

if (password.Length > 0)

{

if (truyen != null)

{

truyen(password);

}

this.Close();

}

else

{

// Hiển thị thông báo lỗi

MessageBox.Show("Hãy nhập mật khẩu!", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

// Kiểm tra trạng thái của checkbox

if (checkBox1.Checked)

{

// Nếu checkbox được tích, đặt thuộc tính PasswordChar của textbox là '\0' để hiển thị các ký tự

PassWork.PasswordChar = '\0';

}

else

{

// Nếu checkbox không được tích, đặt thuộc tính PasswordChar của textbox là '\*' để ẩn các ký tự

PassWork.PasswordChar = '\*';

}

}

}

}

**FORM MAIN**

using AES\_File;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace atbmtt

{

public partial class Form1 : Form

{

string password;

string password1;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

textBox1.Text = "Vui lòng nhập đường dẫn!";

textBox1.ForeColor = Color.Gray;

}

private void button1\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.InitialDirectory = "c:\\";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

textBox1.Text = openFileDialog.FileName;

}

}

private void label1\_Click\_1(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void textBox1\_DragEnter(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.FileDrop))

e.Effect = DragDropEffects.Copy;

}

private void textBox1\_DragDrop(object sender, DragEventArgs e)

{

string[] files = (string[])e.Data.GetData(DataFormats.FileDrop);

if (files.Length > 0) { textBox1.Text = files[0]; }

}

private void textBox1\_Enter\_1(object sender, EventArgs e)

{

if (textBox1.Text == "Vui lòng nhập đường dẫn!")

{

textBox1.Text = "";

textBox1.ForeColor = Color.Black;

}

}

private void textBox1\_Leave\_1(object sender, EventArgs e)

{

if (textBox1.Text == "")

{

textBox1.Text = "Vui lòng nhập đường dẫn!";

textBox1.ForeColor = Color.Gray;

}

}

//Phương thức nhận dữ liệu từ form Pass

public void ReceiveData(string data)

{

//Lưu dữ liệu vào biến passwork

password = data;

}

private void button2\_Click(object sender, EventArgs e)//Sự kiện mã hóa

{

// Lấy đường dẫn từ textbox

string inputFile = textBox1.Text;

// Kiểm tra xem đường dẫn có rỗng không

if (inputFile=="Vui lòng nhập đường dẫn!")

{

// Hiển thị thông báo lỗi

MessageBox.Show("Hãy nhập đường dẫn file!", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

// Tiếp tục các bước mã hóa

//Tạo form con

Pass child = new Pass();

//Gán delegate với phương thức nhận dữ liệu

child.truyen = ReceiveData;// Mật khẩu để mã hóa

//Hiển thị form con

child.ShowDialog();

AES aes = new AES(); // Tạo đối tượng AES

aes.FileEncrypt(inputFile, password); // Gọi phương thức FileEncrypt

// Hiển thị thông báo cho người dùng

MessageBox.Show("Tệp tin đã được mã hóa thành công!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Tạo đường dẫn để lưu tệp tin đã mã hóa// string outputFile = Path.GetDirectoryName(inputFile);

string outputFile = Path.Combine(Path.GetDirectoryName(inputFile), Path.GetFileNameWithoutExtension(inputFile) + ".aes");

// Ghi password vào outputFile

using (StreamWriter sw = new StreamWriter(outputFile, true)) // Mở tệp tin ở chế độ Append

{

sw.WriteLine(password); // Ghi password vào dòng cuối cùng của tệp tin

sw.Close(); // Đóng tệp tin

}

// Hiển thị vị trí của tệp tin đã mã hóa

MessageBox.Show("Tệp tin đã được lưu tại: " + outputFile, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

//Phương thức nhận dữ liệu từ form passdexryption

public void ReceiveData1(string data1)

{

//Lưu dữ liệu vào biến passwork1

password1 = data1;

}

private void button3\_Click(object sender, EventArgs e)//Sự kiện giải mã

{

string inputFile = textBox1.Text; // Lấy đường dẫn từ textbox

// Kiểm tra xem đường dẫn có rỗng không

if (inputFile == "Vui lòng nhập đường dẫn!")

{

// Hiển thị thông báo lỗi

MessageBox.Show("Hãy nhập đường dẫn file!", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

// Giải mã tệp tin bằng lớp AES

//Tạo form con

passdecryption con = new passdecryption();

//Gán delegate với phương thức nhận dữ liệu

con.TruyenPass = ReceiveData1;// Mật khẩu để mã hóa

//Hiển thị form con

con.ShowDialog();

// Mật khẩu để giải mã

// Đọc password từ inputFile

using (StreamReader sr = new StreamReader(inputFile)) // Mở tệp tin ở chế độ Read

{

string line; // Biến lưu trữ mỗi dòng đọc được

string passwordinput = ""; // Biến lưu trữ password

while ((line = sr.ReadLine()) != null) // Đọc từng dòng cho đến khi hết tệp tin

{

passwordinput = line; // Gán password bằng dòng cuối cùng của tệp tin

}

sr.Close(); // Đóng tệp tin

}

// So sánh password với một biến khác

string anotherPassword = password1; // Biến khác để so sánh

if (password == anotherPassword) // Nếu hai biến bằng nhau

{

AES aes = new AES(); // Tạo đối tượng AES

// Tạo đường dẫn để lưu tệp tin đã giải mã

string outputFile = Path.Combine(Path.GetDirectoryName(inputFile), Path.GetFileNameWithoutExtension(inputFile) + "\_decrypted.docx");

// Gọi phương thức FileDecrypt

aes.FileDecrypt(inputFile, outputFile, password);

// Hiển thị thông báo cho người dùng

MessageBox.Show("Tệp tin đã được giải mã thành công!", "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Hiển thị vị trí của tệp tin đã giải mã

MessageBox.Show("Tệp tin đã được lưu tại: " + outputFile, "Thông báo", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else // Nếu hai biến khác nhau

{

MessageBox.Show("Sai mật khẩu!", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

}

}

namespace AES\_File

{

public class AES

{

// Sử dụng hằng số để định nghĩa kích thước khóa, khối và salt

private const int KeySize = 256; // Độ dài khóa AES (tính bằng bit)

private const int BlockSize = 128; // Độ dài khối AES (tính bằng bit)

private const int SaltSize = 32; // Độ dài salt (tính bằng byte)

// Phương thức sinh salt ngẫu nhiên

public static byte[] GenerateRandomSalt()

{

byte[] data = new byte[SaltSize];

using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())

{

rng.GetBytes(data); // Sinh salt ngẫu nhiên

}

return data;

}

/\* Phương thức GenerateRandomSalt có ý nghĩa là sinh ra một salt ngẫu nhiên kiểu mảng byte và trả về kết quả. Salt là dữ liệu ngẫu nhiên được thêm vào mật khẩu để tăng cường an ninh và ngẫu nhiên của khóa hoặc IV mã hóa. Salt làm cho việc băm mật khẩu khó bị phá vỡ bởi các bảng băm trước hoặc các cuộc tấn công bẻ khóa. M thường được lưu trữ cùng với giá trị băm trong cơ sở dữ liệu, và được nối liền với mật khẩu trước khi băm.

Đối tượng kiểu **RNGCryptoServiceProvider** là một lớp được định nghĩa trong .NET Framework, nó cung cấp các chức năng liên quan đến sinh số ngẫu nhiên an toàn cho các ứng dụng mã hóa (cryptography).\*/

// Phương thức mã hóa file

public void FileEncrypt(string inputFile, string password)

{

byte[] salt = GenerateRandomSalt(); // Sinh salt ngẫu nhiên

FileStream fsCrypt = new FileStream(inputFile + ".aes", FileMode.Create); // Tạo file output

byte[] passwordBytes = Encoding.UTF8.GetBytes(password); // Chuyển đổi mật khẩu sang mảng byte

//Encoding.UTF8 là một thuộc tính của không gian tên System.Text trong .NET mà trả về một đối tượng của lớp UTF8Encoding. UTF8Encoding là một hệ thống mã hóa ký tự cho Unicode có thể dịch bất kỳ ký tự Unicode nào thành một chuỗi nhị phân duy nhất tương ứng và ngược lại. UTF-8 tương thích ngược với ASCII và là mã hóa ưa thích cho e-mail và trang web

RijndaelManaged AES = new RijndaelManaged(); // Tạo đối tượng AES

AES.KeySize = KeySize; // Thiết lập độ dài khóa

AES.BlockSize = BlockSize; // Thiết lập độ dài khối

AES.Padding = PaddingMode.PKCS7; // Thiết lập chế độ padding

**/\***Chế độ thêm bớt dữ liệu PKCS #7 là một phương pháp để thêm vào hoặc loại bỏ khỏi dữ liệu một số byte để làm cho kích thước của dữ liệu là bội số của kích thước khối của thuật toán mã hóa. Ví dụ, với AES, kích thước khối là 16 byte, nếu dữ liệu có kích thước là 10 byte, thì sẽ cần thêm 6 byte để làm đầy khối cuối cùng. Các byte được thêm vào có giá trị bằng với số byte cần thêm, ví dụ, 0x06. Khi giải mã, các byte được thêm vào sẽ được loại bỏ khỏi dữ liệu, dựa vào giá trị của byte cuối cùng. [Chế độ thêm bớt dữ liệu PKCS #7 giúp quá trình mã hóa và giải mã dữ liệu được chính xác và an toàn](https://stackoverflow.com/questions/28592989/aes-pkcs7-padding) \*/

var key = new Rfc2898DeriveBytes(passwordBytes, salt, 50000); // Tạo khóa từ mật khẩu và salt

Lớp Rfc2898DeriveBytes: đây là một cách để tạo ra một khóa mật mã từ một mật khẩu và một salt. Salt là một giá trị ngẫu nhiên được thêm vào mật khẩu để làm cho nó khó hơn cho kẻ tấn công đoán hoặc phá vỡ mật khẩu. Lớp Rfc2898DeriveBytes thực hiện một thuật toán tiêu chuẩn gọi là PBKDF2, viết tắt của Password-Based Key Derivation Function 2. Thuật toán này lặp đi lặp lại băm mật khẩu và muối bằng một hàm gọi là HMACSHA1, đây là một loại thuật toán băm an toàn sử dụng một khóa bí mật. Số lần lặp (50000 trong trường hợp này) xác định số lần băm được thực hiện, và cũng ảnh hưởng đến an ninh và hiệu suất của việc tạo khóa. Càng nhiều lần lặp, khóa càng an toàn, nhưng cũng càng tốn thời gian và tài nguyên để tạo ra nó. Phương thức GetBytes trả về một mảng byte có kích thước được chỉ định có thể được sử dụng như một khóa hoặc vector khởi tạo (IV) cho mã hóa hoặc giải mã.

Mục đích của việc sử dụng lớp này thay vì sử dụng trực tiếp mật khẩu như là một khóa hoặc IV là để tăng cường an ninh và ngẫu nhiên của khóa hoặc IV. Một mật khẩu riêng có thể không đủ dài, phức tạp hoặc ngẫu nhiên để làm một khóa hoặc IV mã hóa tốt. Bằng cách sử dụng muối và băm, lớp Rfc2898DeriveBytes có thể tạo ra các khóa hoặc IV có bất kỳ độ dài nào và có nhiều entropy (ngẫu nhiên) hơn so với chỉ có mật khẩu. Điều này khiến cho việc đoán, bẻ khóa hoặc đảo ngược khóa hoặc IV từ mật khẩu hoặc dữ liệu đã mã hóa trở nên khó khăn hơn cho kẻ tấn công.

AES.Key = key.GetBytes(KeySize / 8); // Lấy khóa bí mật

AES.IV = key.GetBytes(BlockSize / 8); // Lấy vector khởi tạo

AES.Mode = CipherMode.CFB; // Thiết lập chế độ mã hóa

**/\*CipherMode.CFB**, là một hằng số được định nghĩa trong lớp **CipherMode**, nó đại diện cho chế độ mã hóa phản hồi khối (cipher feedback mode).

Chế độ mã hóa phản hồi khối (CFB) là một chế độ xử lý các đoạn nhỏ của văn bản gốc thành văn bản mã, thay vì xử lý một khối hoàn chỉnh một lúc. Chế độ này sử dụng một thanh ghi dịch (shift register) có độ dài bằng một khối và được chia thành các phần. [Ví dụ, nếu kích thước khối là 8 byte, với mỗi byte được xử lý một lần, thanh ghi dịch sẽ được chia thành tám phần](https://www.demo2s.com/csharp/csharp-ciphermode-cfb.html). Nếu một bit trong văn bản mã bị biến dạng, một bit văn bản gốc sẽ bị biến dạng và thanh ghi dịch sẽ bị hỏng. Điều này dẫn đến các đoạn văn bản gốc tiếp theo sẽ bị biến dạng cho đến khi bit xấu được dịch ra khỏi thanh ghi dịch. Kích thước phản hồi (feedback size) mặc định có thể thay đổi theo thuật toán, nhưng thường là 8 bit hoặc số bit của kích thước khối.

Câu lệnh **AES.Mode = CipherMode.CFB;** có ý nghĩa là thiết lập chế độ mã hóa phản hồi khối cho đối tượng **AES**, để mã hóa và giải mã dữ liệu theo cách an toàn và linh hoạt hơn so với chế độ mã hóa sổ sách điện tử (ECB) hay chế độ mã hóa chuỗi khối (CBC). \*/

fsCrypt.Write(salt, 0, salt.Length); // Ghi salt vào file output

CryptoStream cs = new CryptoStream(fsCrypt, AES.CreateEncryptor(), CryptoStreamMode.Write); // Tạo luồng mã hóa

//Câu lệnh **CryptoStream cs = new CryptoStream(fsCrypt, AES.CreateEncryptor(), CryptoStreamMode.Write);** có ý nghĩa là khởi tạo một luồng mã hóa để ghi dữ liệu đã mã hóa theo thuật toán AES vào luồng đích fsCrypt. Câu lệnh này có thể được sử dụng để bảo vệ dữ liệu khi lưu trữ hoặc truyền tải.

FileStream fsIn = new FileStream(inputFile, FileMode.Open); // Mở file input

byte[] buffer = new byte[1048576]; // Tạo bộ đệm để đọc và ghi file

**/\* byte[] buffer = new byte[1048576];**: Đây là một câu lệnh trong ngôn ngữ lập trình C#, nó khai báo và khởi tạo một biến kiểu mảng byte có tên là **buffer**, kích thước của mảng là **1048576** phần tử. Mỗi phần tử trong mảng có kiểu byte, là một kiểu dữ liệu nguyên thủy trong C#, nó có kích thước là 8 bit và có thể lưu trữ các giá trị từ 0 đến 255. Câu lệnh này cấp phát bộ nhớ cho biến **buffer** và gán giá trị mặc định là 0 cho tất cả các phần tử trong mảng. \*/

int read;

try

{

while ((read = fsIn.Read(buffer, 0, buffer.Length)) > 0) // Đọc file input cho đến khi hết

{

Application.DoEvents();

cs.Write(buffer, 0, read); // Ghi file output đã mã hóa

}

fsIn.Close(); // Đóng file input

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

//không đọc được file sẽ hiện thông báo lỗi

} finally

{

cs.Close(); // Đóng luồng mã hóa

fsCrypt.Close(); // Đóng file output

key.Dispose(); // Xóa khóa khỏi bộ nhớ

AES.Clear(); // Xóa các thông số của AES khỏi bộ nhớ

}

}

// Phương thức giải mã file

public void FileDecrypt(string inputFile, string outputFile, string password)

{

byte[] passwordBytes = Encoding.UTF8.GetBytes(password); // Chuyển đổi mật khẩu sang mảng byte

byte[] salt = new byte[SaltSize]; // Tạo mảng byte để lưu salt

FileStream fsCrypt = new FileStream(inputFile, FileMode.Open); // Mở file input

fsCrypt.Read(salt, 0, salt.Length); // Đọc salt từ file input

RijndaelManaged AES = new RijndaelManaged(); // Tạo đối tượng AES

AES.KeySize = KeySize; // Thiết lập độ dài khóa

AES.BlockSize = BlockSize; // Thiết lập độ dài khối

var key = new Rfc2898DeriveBytes(passwordBytes, salt, 50000); // Tạo khóa từ mật khẩu và salt

AES.Key = key.GetBytes(KeySize / 8); // Lấy khóa bí mật

AES.IV = key.GetBytes(BlockSize / 8); // Lấy vector khởi tạo

AES.Padding = PaddingMode.PKCS7;// Thiết lập chế độ padding

AES.Mode = CipherMode.CFB;

CryptoStream cs = new CryptoStream(fsCrypt, AES.CreateDecryptor(), CryptoStreamMode.Read);

FileStream fsOut = new FileStream(outputFile, FileMode.Create);

int read;

byte[] buffer = new byte[1048576];

try

{

while ((read = cs.Read(buffer, 0, buffer.Length)) > 0)

{

Application.DoEvents();

fsOut.Write(buffer, 0, read);

}

fsOut.Close();//đóng luồng dữ liệu fsout và giải phóng tài nguyên liên quan key.Dispose();// Xóa khóa khỏi bộ nhớ

AES.Clear();// Xóa các thông số của AES khỏi bộ nhớ

File.Delete(inputFile);

}

catch (CryptographicException ex\_CryptographicException)

{

Console.WriteLine("CryptographicException error: " + ex\_CryptographicException.Message);

}

catch (Exception ex)

{

Console.WriteLine("Error: " + ex.Message);

}

try

{

cs.Close();

}

catch (Exception ex)

{

Console.WriteLine("Error by closing CryptoStream: " + ex.Message);

}

finally

{

fsCrypt.Close();

}

}

}

}

**TỔNG KẾT**

Chúng tôi đã giới thiệu về chương trình, cách hoạt động, và thuật toán được sử dụng. Hi vọng qua bài báo cáo này bạn sẽ hiểu được đôi chút về tầm quan trọng của việc bảo mật thông tin cá nhân trong thời đại hiện nay. Nhưng không có gì là hoàn hảo cả, chương trình của chúng tôi cũng có những vấn đề đã giải quyết được và những vẫn đề còn tồn đọng.

Vấn đề đã được giải quyết: thiết kế được 1 giao diện đơn giản thân thiện với người dùng, giải quyết được vấn đề mã hóa và giải mã tệp tin.

Vấn đề còn tồn đọng: hiện tại chương trình chỉ mã hóa được file Work có định dạng .docx, khi nhấn mã hóa sẽ tạo ra 2 file mã hóa.

Trên đây là tất cả những gì chúng tôi đã làm để xây dựng lên 1 chương trình mã hóa để giúp người dùng có thể tự bảo mật những thông tin quan trọng của họ trong thời đại công nghệ 4.0 hiện nay.