using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

namespace AESalgo

{

class Program

{

static void Main(string[] args)

{

// Set up the key and IV for encryption

byte[] key = Encoding.UTF8.GetBytes("abcdefghijklmnop");

byte[] iv = Encoding.UTF8.GetBytes("1234567890123456");

// Create an instance of the AesEncryptor class

AesEncryptor encryptor = new AesEncryptor(key, iv);

// Define some sample JSON data to encrypt

string jsonData = "{\"name\":\"John Doe\",\"age\":30,\"city\":\"New York\"}";

Console.WriteLine("Original JSON data: " + jsonData);

// Convert the JSON data to a byte array

byte[] jsonBytes = Encoding.UTF8.GetBytes(jsonData);

// Encrypt the JSON data

byte[] encryptedData = encryptor.Encrypt(jsonBytes);

// Print out the encrypted data as a Base64-encoded string

string encryptedString = Convert.ToBase64String(encryptedData);

Console.WriteLine("Encrypted JSON data: " + encryptedString);

// Decrypt the encrypted data

byte[] decryptedData = encryptor.Decrypt(encryptedData);

// Convert the decrypted data back to a string

string decryptedString = Encoding.UTF8.GetString(decryptedData);

Console.WriteLine("Decrypted JSON data: " + decryptedString);

// Parse the decrypted JSON data and print out its contents

JsonDocument parsedJson = JsonDocument.Parse(decryptedString);

Console.WriteLine("Parsed JSON data:");

foreach (JsonProperty property in parsedJson.RootElement.EnumerateObject())

{

Console.WriteLine(property.Name + ": " + property.Value);

}

}

}

public class AesEncryptor

{

private byte[] key;

private byte[] iv;

public AesEncryptor(string key)

{

if (key == null)

{

throw new ArgumentNullException("key");

}

if (key.Length != 16 && key.Length != 24 && key.Length != 32)

{

throw new ArgumentException("Invalid key size. Key must be 16, 24, or 32 bytes long.");

}

this.key = Encoding.UTF8.GetBytes(key);

}

public AesEncryptor(string key, string iv) : this(key)

{

if (iv == null)

{

throw new ArgumentNullException("iv");

}

if (iv.Length != 16)

{

throw new ArgumentException("Invalid IV size. IV must be 16 bytes long.");

}

this.iv = Encoding.UTF8.GetBytes(iv);

}

public byte[] Encrypt(string json)

{

if (json == null)

{

throw new ArgumentNullException("json");

}

byte[] input = Encoding.UTF8.GetBytes(json);

using (Aes aes = Aes.Create())

{

aes.Key = this.key;

if (this.iv != null)

{

aes.IV = this.iv;

}

else

{

aes.GenerateIV();

this.iv = aes.IV;

}

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, aes.CreateEncryptor(), CryptoStreamMode.Write))

{

cs.Write(input, 0, input.Length);

}

byte[] encrypted = ms.ToArray();

byte[] result = new byte[this.iv.Length + encrypted.Length];

Buffer.BlockCopy(this.iv, 0, result, 0, this.iv.Length);

Buffer.BlockCopy(encrypted, 0, result, this.iv.Length, encrypted.Length);

return result;

}

}

}

public string Decrypt(byte[] input)

{

if (input == null)

{

throw new ArgumentNullException("input");

}

if (input.Length < 16)

{

throw new ArgumentException("Invalid input length.");

}

byte[] iv = new byte[16];

Buffer.BlockCopy(input, 0, iv, 0, 16);

byte[] encrypted = new byte[input.Length - 16];

Buffer.BlockCopy(input, 16, encrypted, 0, encrypted.Length);

using (Aes aes = Aes.Create())

{

aes.Key = this.key;

aes.IV = iv;

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, aes.CreateDecryptor(), CryptoStreamMode.Write))

{

cs.Write(encrypted, 0, encrypted.Length);

}

byte[] decryptedBytes = ms.ToArray();

return Encoding.UTF8.GetString(decryptedBytes);

}

}

}

}

}