Cryptography in C#

**Introduction to Cryptography in C#**

Cryptography ensures secure communication and data storage. In C#, you can implement cryptographic algorithms like hashing, symmetric encryption, and asymmetric encryption.

**1. Hashing (SHA256 Example)**

Hashing generates a fixed-size output from input data. It is commonly used for data integrity checks or storing passwords securely.

using System.Security.Cryptography;  
using System.Text;  
  
string data = "mySensitiveData";  
using (SHA256 sha256 = SHA256.Create())  
{  
 byte[] bytes = Encoding.UTF8.GetBytes(data);  
 byte[] hash = sha256.ComputeHash(bytes);  
  
 StringBuilder hashString = new StringBuilder();  
 foreach (byte b in hash)  
 {  
 hashString.Append(b.ToString("x2"));  
 }  
  
 Console.WriteLine($"Hashed value: {hashString}");  
}

**2. Symmetric Encryption (AES Example)**

Symmetric encryption uses a single key for both encryption and decryption. AES is a popular symmetric algorithm.

\*\*AES Encryption Example:\*\*

using System;  
using System.IO;  
using System.Security.Cryptography;  
using System.Text;  
  
string plainText = "Sensitive information";  
  
using (Aes aes = Aes.Create())  
{  
 aes.Key = Encoding.UTF8.GetBytes("A very strong key!!");  
 aes.IV = Encoding.UTF8.GetBytes("A strong IV text!");  
  
 ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);  
  
 using (MemoryStream ms = new MemoryStream())  
 {  
 using (CryptoStream cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))  
 {  
 using (StreamWriter sw = new StreamWriter(cs))  
 {  
 sw.Write(plainText);  
 }  
 byte[] encrypted = ms.ToArray();  
 Console.WriteLine($"Encrypted: {Convert.ToBase64String(encrypted)}");  
 }  
 }  
}

\*\*AES Decryption Example:\*\*

string encryptedText = "Encrypted data here";  
byte[] cipherText = Convert.FromBase64String(encryptedText);  
  
using (Aes aes = Aes.Create())  
{  
 aes.Key = Encoding.UTF8.GetBytes("A very strong key!!");  
 aes.IV = Encoding.UTF8.GetBytes("A strong IV text!");  
  
 ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key, aes.IV);  
  
 using (MemoryStream ms = new MemoryStream(cipherText))  
 {  
 using (CryptoStream cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read))  
 {  
 using (StreamReader sr = new StreamReader(cs))  
 {  
 string decrypted = sr.ReadToEnd();  
 Console.WriteLine($"Decrypted: {decrypted}");  
 }  
 }  
 }  
}

**3. Asymmetric Encryption (RSA Example)**

Asymmetric encryption uses a pair of keys, one public and one private. The public key encrypts the data, while the private key decrypts it.

\*\*RSA Encryption Example:\*\*

using System;  
using System.Security.Cryptography;  
using System.Text;  
  
string data = "My secret data";  
  
using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())  
{  
 byte[] dataBytes = Encoding.UTF8.GetBytes(data);  
 byte[] encrypted = rsa.Encrypt(dataBytes, false);  
  
 Console.WriteLine($"Encrypted: {Convert.ToBase64String(encrypted)}");  
}

\*\*RSA Decryption Example:\*\*

string encryptedData = "Encrypted base64 string";  
byte[] encryptedBytes = Convert.FromBase64String(encryptedData);  
  
using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())  
{  
 byte[] decrypted = rsa.Decrypt(encryptedBytes, false);  
 Console.WriteLine($"Decrypted: {Encoding.UTF8.GetString(decrypted)}");  
}

**4. Digital Signatures**

Digital signatures provide a way to verify the authenticity and integrity of data using asymmetric cryptography.

\*\*Creating a Digital Signature Example:\*\*

using System;  
using System.Security.Cryptography;  
using System.Text;  
  
string data = "Important document";  
  
using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())  
{  
 byte[] dataBytes = Encoding.UTF8.GetBytes(data);  
 byte[] signature = rsa.SignData(dataBytes, CryptoConfig.MapNameToOID("SHA256"));  
  
 Console.WriteLine($"Signature: {Convert.ToBase64String(signature)}");  
}

\*\*Verifying a Digital Signature Example:\*\*

string originalData = "Important document";  
byte[] signatureBytes = Convert.FromBase64String("Base64SignatureHere");  
  
using (RSACryptoServiceProvider rsa = new RSACryptoServiceProvider())  
{  
 byte[] originalBytes = Encoding.UTF8.GetBytes(originalData);  
 bool isVerified = rsa.VerifyData(originalBytes, CryptoConfig.MapNameToOID("SHA256"), signatureBytes);  
  
 Console.WriteLine($"Signature is valid: {isVerified}");  
}