UNIWERSTYTET KAZIMIERZA WIELKIEGO
W BYDGOSZCZY

Szyfrowanie symetryczne i asymetryczne

Bezpieczeństwo systemów komputerowych

Łukasz Pietraszek

III rok informatyka

Spis treści

1. Zastosowanie szyfrowania symetrycznego oraz asymetrycznego

1.1. Szyfrowanie symetryczne, a szyfrowanie asymetryczne

Algorytmy szyfrowania podobnie jak kryptografia dzielą się na dwie kategorie: algorytmy szyfrowania symetrycznego i algorytmy szyfrowania asymetrycznego. Podstawowa różnica między tymi dwiema metodami szyfrowania polega na tym, że algorytmy szyfrowania symetrycznego opierają się na pojedynczym klucz, podczas gdy szyfrowanie asymetryczne polega na wykorzystaniu dwóch różnych, ale powiązanych ze sobą kluczy. Takie rozróżnienie, choć pozornie proste, najlepiej wyjaśnia różnice funkcjonalne występujące między tymi dwiema formami szyfrowania oraz sposobem ich wykorzystania.

1.2. Klucze szyfrujące

W kryptografii algorytmy szyfrowania generują klucze, które są niczym innym jak serią bitów używanych do szyfrowania i deszyfrowania informacji. Sposób wykorzystania tych kluczy zależy od wybranej metody szyfrowania.

Podczas gdy algorytmy szyfrowania symetrycznego wykorzystują ten sam klucz do wykonywania zarówno funkcji szyfrowania, jak i deszyfrowania, algorytm szyfrowania asymetrycznego wykorzystuje jeden klucz do szyfrowania danych, a inny klucz do ich odszyfrowania. W systemach asymetrycznych pierwszym kluczem jest ten, który jest wykorzystywany do szyfrowania danych. Klucz ten nazywany jest kluczem publicznym. Jak wskazuje już sama nazwa, można bez żadnych obaw udostępniać go innym osobom. Drugim kluczem natomiast jest klucz prywatny, który służy do deszyfrowania danych. Dostęp do tego klucza powinien mieć tylko jego właściciel.

1.3. Przykładowe zastosowania

a) Szyfrowanie symetryczne

Ze względu na większą szybkość, szyfrowanie symetryczne jest szeroko stosowane do ochrony informacji w wielu nowoczesnych systemach komputerowych. Najbardziej znanymi przykładami są m.in *Advanced Encryption Standard (AES)* wykorzystywany przez rząd Stanów Zjednoczonych do szyfrowania poufnych i tajnych informacji. *AES* zastąpił wcześniejszy standard szyfrowania danych (*DES*), który został opracowany w latach siedemdziesiątych jako jeden ze standardów szyfrowania symetrycznego.

b) Szyfrowanie asymetryczne

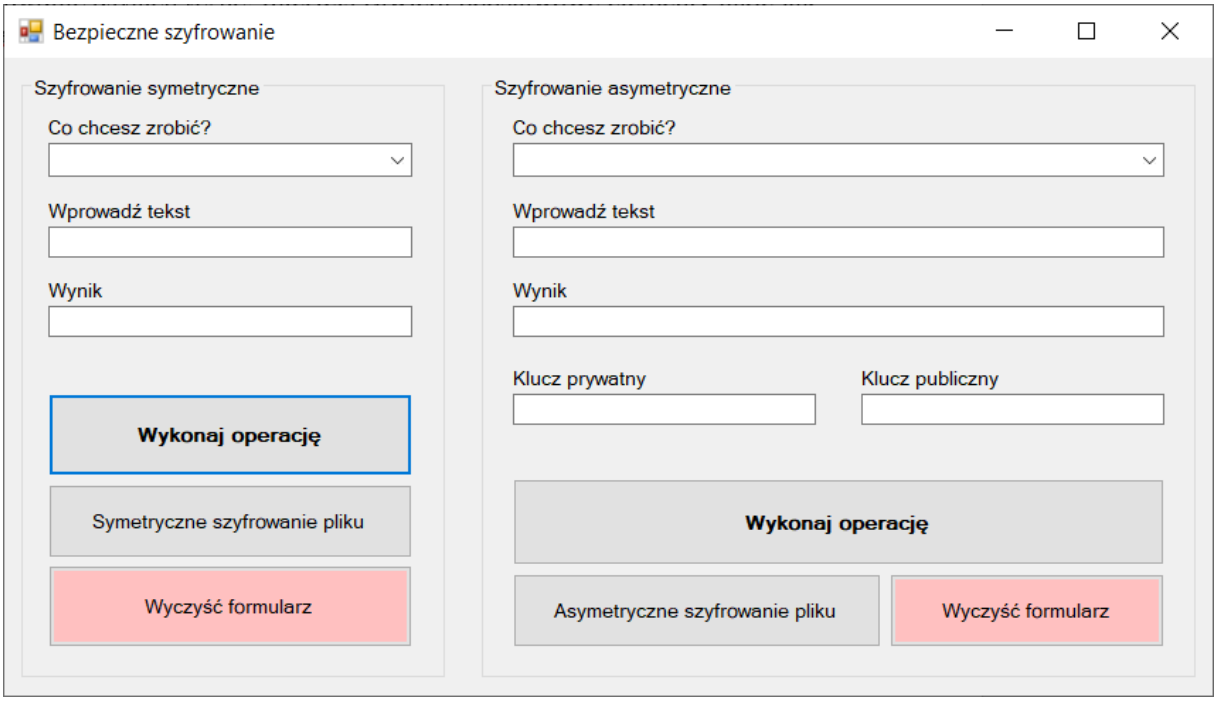
Szyfrowanie asymetryczne z kolei jest wykorzystywane w systemach, w których wielu użytkowników może wymagać dostępu do zarówno szyfrowania, jak i deszyfrowania wiadomości lub zestawu danych. Jednym z przykładów takiego systemu jest zaszyfrowana wiadomość e-mail, w której klucz publiczny może być użyty do zaszyfrowania wiadomości, a klucz prywatny może zostać użyty do jej odszyfrowania.

c) Szyfrowanie hybrydowe

W wielu obecnie dostępnych na rynku aplikacjach wykorzystuje się już zarówno szyfrowanie symetryczne jak i asymetryczne. Typowymi przykładami takich systemów hybrydowych są protokoły *Security Sockets Layer (SSL)* i *Transport Layer Security (TLS)*, które zaprojektowano w celu zapewnienia bezpiecznej komunikacji w sieci Internet.

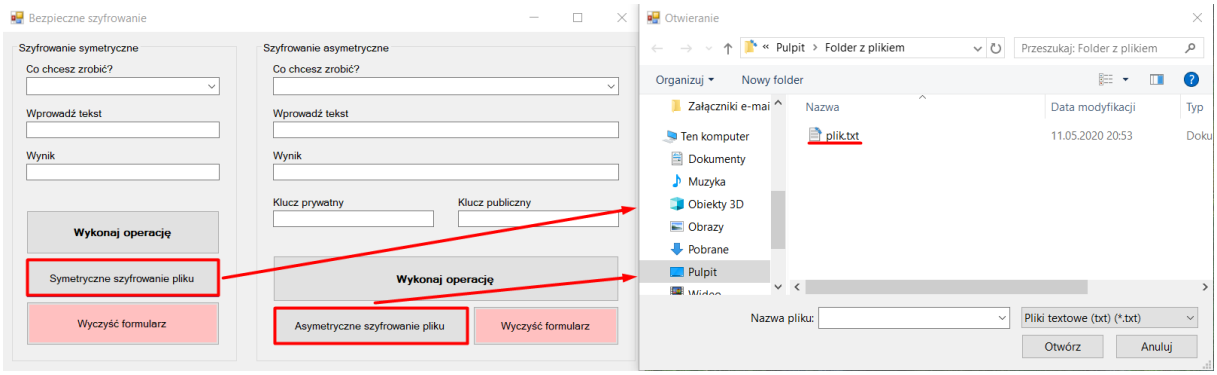
2. Graficzna część aplikacji

Aplikacja została podzielona na dwa niezależne segmenty (elementy *groupBox*): Szyfrowanie symetryczne oraz Szyfrowanie asymetryczne. Interfejs zawiera podstawowe elementy takie jak *comboBox (wartości „zaszyfruj" i „odszyfruj", textBox, label, button.*



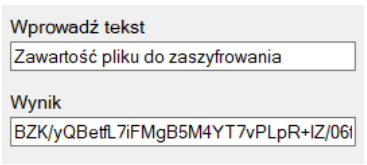
Rysunek 1. Wygląd aplikacji

Wczytanie tekstu do zaszyfrowania jest możliwe zarówno wpisując frazę w pole „Wprowadź tekst", jak i poprzez załączenie pliku tekstowego (.txt). W celu podpięcia pliku używany jest Eksplorator plików systemu Windows.



Rysunek 2. Wybieranie pliku w eksploratorze plików

Zawartość pliku wczytuje się w pole „Wprowadź tekst", a w polu „Wynik" efekt szyfrowania.



Rysunek 3. Zaszyfrowana zawartość pliku

3. Kod aplikacji

3.1. Obsługa elementów kontrolek Windows Forms dla szyfrowania symetrycznego

```
#region --- SZYFROWANIE SYMETRYCZNE ---
#region Wykonaj operację
private void buttonSym_Click(object sender, EventArgs e)
{
    if (comboBoxSym.SelectedIndex < 0)                                          //walidacja comboBox
    {
        MessageBox.Show("Wybierz wartość dla pola comboBox 'Co chcesz zrobić?'");
    }
    else if (string.IsNullOrWhiteSpace(textBoxInputSym.Text))                   //walidacja textBox
    {
        MessageBox.Show("Wypełnij pole 'Wprowadź tekst'");
    }
    else
    {
        string textSym = textBoxInputSym.Text;                                  //wczytanie wartości z textBoxa
        string password = "zaq1@WSX";                                           //stworzenie hasła
        byte[] Salt = Symmetric.GenerateSalt();                                 //wygenerowanie soli
        byte[] IV = Symmetric.GenerateIV();                                     //wygenerowanie IV
        byte[] key = Symmetric.CreateKey(password, Salt);                       //stworzenie klucza

        string Encrypted = Symmetric.EncryptString(textSym, key, IV);           //funkcja szyfrowania
        string Decrypted = Symmetric.DecryptString(Encrypted, key, IV);         //funkcja deszyfrowania

        if (comboBoxSym.SelectedIndex == 1) { textBoxResultSym.Text = Decrypted; }  //odszyfrowywanie
        if (comboBoxSym.SelectedIndex == 0) { textBoxResultSym.Text = Encrypted; }  //zaszyfrowywanie
    }
}
#endregion
```

Rysunek 4. SZYFROWANIE SYMETRYCZNE -> Wykonaj operację

```
#region Symetryczne szyfrowanie pliku
private void buttonSymFile_Click(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();
    dialog.Filter = "Pliki textowe (txt)|*.txt";                    //połączenie z plikiem
    if (dialog.ShowDialog() == DialogResult.OK)                     //sprawdzenie połączenia
    {
        textBoxInputSym.Text = File.ReadAllText(dialog.FileName);   //przypisanie wartości z pliku do textBoxa "Wprowadź tekst"
        string toEncrypt = File.ReadAllText(dialog.FileName);       //przypisanie wartości z pliku do zmiennej
        string password = "zaq1@WSX";                               //stworznie hasła
        byte[] Salt = Symmetric.GenerateSalt();                     //wygenerowanie soli
        byte[] IV = Symmetric.GenerateIV();                         //wygenerowanie IV
        byte[] key = Symmetric.CreateKey(password, Salt);           //stworzenie klucza
        string Encrypted = Symmetric.EncryptString(toEncrypt, key, IV); //zaszyfrowanie treści
        textBoxResultSym.Text = Encrypted;                          //przypisanie wyniku
    }
}
#endregion
```

Rysunek 5. SZYFROWANIE SYMETRYCZNE -> Symetryczne szyfrowanie pliku

```
#region Wyczyść formularz
private void buttonSymClear_Click(object sender, EventArgs e)
{
    textBoxInputSym.Clear();        //czyszczenie textBoxa "Wprowadź text"
    textBoxResultSym.Clear();       //czyszczenie textBoxa "Wynik"
    comboBoxSym.ResetText();        //czyszczenie wartości comboBoxa
}
#endregion
#endregion
```

Rysunek 6. SZYFROWANIE SYMETRYCZNIE -> Wyczyść formularz

3.2. Obsługa elementów kontrolek Windows Forms dla szyfrowania symetrycznego

```
#region --- SZYFROWANIE ASYMETRYCZNE ---
#region Wykonaj operację
private void buttonAsym_Click(object sender, EventArgs e)
{
    if (comboBoxAsym.SelectedIndex < 0)                                          //walidacja textBox
    {
        MessageBox.Show("Wybierz wartość dla pola comboBox 'Co chcesz zrobić?'");
    }
    else if (string.IsNullOrWhiteSpace(textBoxInputAsym.Text))                   //walidacja comboBox
    {
        MessageBox.Show("Wypełnij pole 'Wprowadź tekst'");
    }
    else
    {
        var cryptoServiceProvider = new RSACryptoServiceProvider(2048);          //długość klucza
        var privateKey = cryptoServiceProvider.ExportParameters(true);           //klucz prywatny
        var publicKey = cryptoServiceProvider.ExportParameters(false);           //klucz publiczny

        string publicKeyString = Asymmetric.GetKeyString(publicKey);            //przypisanie klucza publicznego do stringa
        string privateKeyString = Asymmetric.GetKeyString(privateKey);           //przypisanie klucza prywatnego do stringa

        textBoxPublicKey.Text = publicKeyString;                                 //wypisanie klucza publicznego w textBoxie
        textBoxPrivateKey.Text = privateKeyString;                               //wypisanie klucza prywatnego w textBoxie
        string inputText = textBoxInputAsym.Text;                                //przypisanie zawartości textBoxa do zmiennej
        string encryptedText = Asymmetric.Encrypt(inputText, publicKeyString);   //funkcja szyfrowania
        string decryptedText = Asymmetric.Decrypt(encryptedText, privateKeyString); //funkcja deszyfrowania

        if (comboBoxAsym.SelectedIndex == 0) { textBoxResultAsym.Text = encryptedText; }
        if (comboBoxAsym.SelectedIndex == 1) { textBoxResultAsym.Text = decryptedText; }
    }
}
#endregion
```

Rysunek 7. SZYFROWANIE ASYMETRYCZNE -> Wykonaj operację

```
#region Asymetryczne szyfrowanie pliku
private void buttonAsymFile_Click(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();
    dialog.Filter = "Pliki textowe (txt)|*.txt";                        //połączenie z plikiem
    if (dialog.ShowDialog() == DialogResult.OK)                         //sprawdzenie połączenia
    {
        textBoxInputAsym.Text = File.ReadAllText(dialog.FileName);      //przypisanie wartości z pliku do textBoxa "Wprowadź tekst"
        string toEncrypt = File.ReadAllText(dialog.FileName);           //przypisanie wartości z pliku do zmiennej
        var cryptoServiceProvider = new RSACryptoServiceProvider(2048); //długość klucza
        var privateKey = cryptoServiceProvider.ExportParameters(true);  //klucz prywatny
        var publicKey = cryptoServiceProvider.ExportParameters(false);  //klucz publiczny

        string publicKeyString = Asymmetric.GetKeyString(publicKey);    //przypisanie klucza publicznego do stringa
        string privateKeyString = Asymmetric.GetKeyString(privateKey);  //przypisanie klucza prywatnego do stringa

        textBoxPublicKey.Text = publicKeyString;                        //wypisanie klucza publicznego w textBoxie
        textBoxPrivateKey.Text = privateKeyString;                      //wypisanie klucza prywatnego w textBoxie
        string encryptedText = Asymmetric.Encrypt(toEncrypt, publicKeyString);
        textBoxResultAsym.Text = encryptedText;                         //wypisywanie zaszyfrowanego wyniku w textBoxie
    }
}
#endregion
```

Rysunek 8. SZYFROWANIE ASYMETRYCZNE -> Asymetryczne szyfrowanie pliku

```
#region Czyszczenie formularza
private void buttonAsymClear_Click(object sender, EventArgs e)
{
    textBoxInputAsym.Clear();                   //czyszczenie textBoxa "Wprowadź text"
    textBoxResultAsym.Clear();                  //czyszczenie textBoxa "Wynik"
    textBoxPublicKey.Clear();                   //czyszczenie textBoxa "Klucz publiczny"
    textBoxPrivateKey.Clear();                  //czyszczenie textBoxa "Klucz prywatny"
    comboBoxAsym.ResetText();                   //czyszczenie wartości comboBoxa
}
#endregion
#endregion
```

Rysunek 9. SZYFROWANIE ASYMETRYCZNE -> Czyszczenie formularza

3.3. Funkcje symetryczne

```
public static string EncryptString(string input, byte[] key, byte[] IV)
{
    using (RijndaelManaged RMCrypto = new RijndaelManaged())
    {
        RMCrypto.Key = key;
        RMCrypto.IV = IV;
        var encryptor = RMCrypto.CreateEncryptor(RMCrypto.Key, RMCrypto.IV);
        var msEncrypt = new MemoryStream();
        var csEncrypt = new CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write);
        using (var swEncrypt = new StreamWriter(csEncrypt))
        {
            swEncrypt.Write(input);
        }
        return Convert.ToBase64String(msEncrypt.ToArray());
    }
}
```

Rysunek 10. Szyfrowanie wprowadzonej zawartości

```
public static string DecryptString(string input, byte[] key, byte[] IV)
{
    using (RijndaelManaged RMCrypto = new RijndaelManaged())
    {
        string text;
        RMCrypto.Key = key;
        RMCrypto.IV = IV;
        var decryptor = RMCrypto.CreateDecryptor(RMCrypto.Key, RMCrypto.IV);
        var cipher = Convert.FromBase64String(input);
        var msDecrypt = new MemoryStream(cipher);
        var csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read);
        using (var srDecrypt = new StreamReader(csDecrypt))
        {
            text = srDecrypt.ReadToEnd();
        }
        return text;
    }
}
```

Rysunek 11. Odszyfrowywanie wprowadzonej zawartości

```
public static byte[] GenerateSalt()
{
    List<byte> Salt = new List<byte>();
    Random rnd = new Random();
    for (int i = 1; i <= 256; i++)
    {
        Salt.Add(Convert.ToByte(rnd.Next(1, 128)));
    }
    return Salt.ToArray();
}
```

Rysunek 12. Generowanie soli

```
public static byte[] GenerateIV()
{
    List<byte> IV = new List<byte>();
    Random rnd = new Random();
    for (int i = 1; i <= 16; i++)
    {
        IV.Add(Convert.ToByte(rnd.Next(0, 128)));
    }
    return IV.ToArray();
}
```

Rysunek 13. Generowanie IV

```
public static byte[] CreateKey(string InputKey, byte[] saltBytes)
{
    using (var key = new Rfc2898DeriveBytes(InputKey, saltBytes, 1200000))
    {
        return key.GetBytes(32);
    }
}
```

Rysunek 14. Tworzenie klucza

```
public static byte[] GetHMAC(byte[] hashdata, byte[] key)
{
    HMACSHA512 hmac = new HMACSHA512();
    hmac.Key = key;
    return hmac.ComputeHash(hashdata);
}
```

Rysunek 15. Hashowanie danych

3.4. Funkcje asymetryczne

```
public static string GetKeyString(RSAParameters publicKey)
{
    var stringWriter = new System.IO.StringWriter();
    var xmlSerializer = new System.Xml.Serialization.XmlSerializer(typeof(RSAParameters));
    xmlSerializer.Serialize(stringWriter, publicKey);
    return stringWriter.ToString();
}
```

Rysunek 16. Pobieranie klucza

```
public static string Encrypt(string textToEncrypt, string publicKeyString)
{
    var bytesToEncrypt = Encoding.UTF8.GetBytes(textToEncrypt);

    using (var rsa = new RSACryptoServiceProvider(2048))
    {
        try
        {
            rsa.FromXmlString(publicKeyString.ToString());
            var encryptedData = rsa.Encrypt(bytesToEncrypt, true);
            var base64Encrypted = Convert.ToBase64String(encryptedData);
            return base64Encrypted;
        }
        finally
        {
            rsa.PersistKeyInCsp = false;
        }
    }
}
```

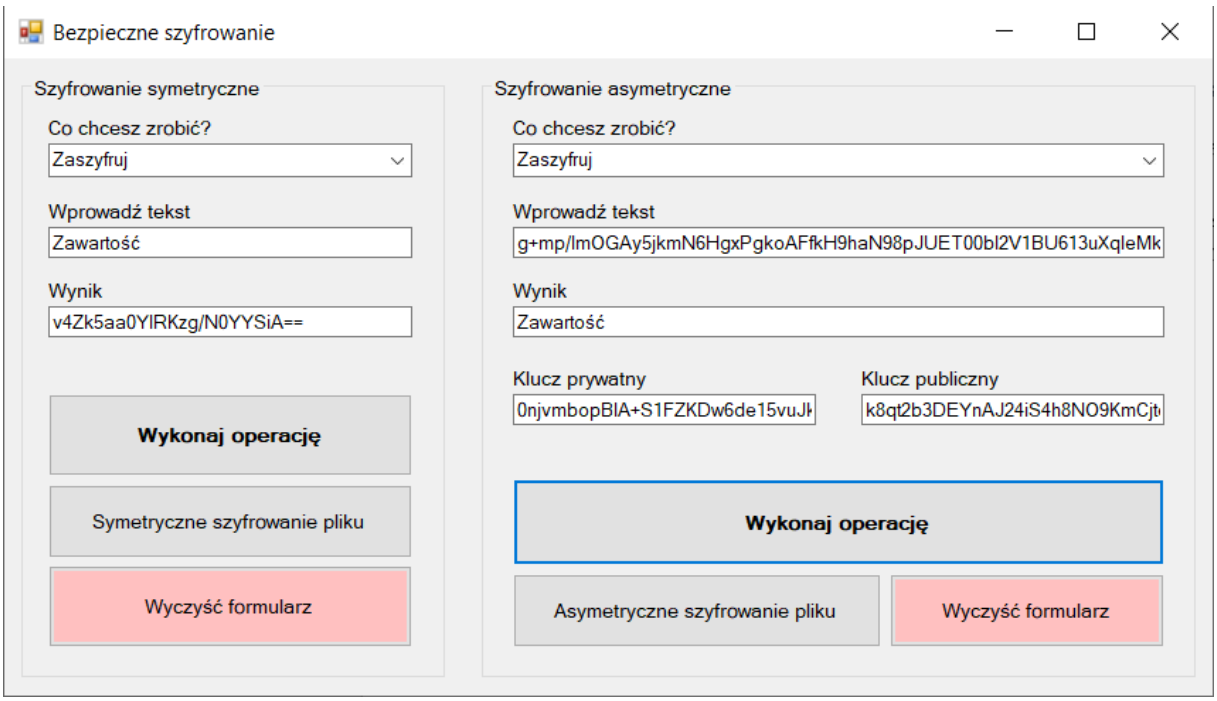
Rysunek 17. Szyfrowanie asymetryczne

```
public static string Decrypt(string textToDecrypt, string privateKeyString)
{
    var bytesToDescrypt = Encoding.UTF8.GetBytes(textToDecrypt);

    using (var rsa = new RSACryptoServiceProvider(2048))
    {
        try
        {
            rsa.FromXmlString(privateKeyString);
            var resultBytes = Convert.FromBase64String(textToDecrypt);
            var decryptedBytes = rsa.Decrypt(resultBytes, true);
            var decryptedData = Encoding.UTF8.GetString(decryptedBytes);
            return decryptedData.ToString();
        }
        finally
        {
            rsa.PersistKeyInCsp = false;
        }
    }
}
```

Rysunek 18. Deszyfrowanie asymetryczne

4. Działanie aplikacji



Rysunek 19. Wykonane szyfrowanie symetryczne i deszyfrowanie asymetryczne

Szyfrowanie symetryczne jest prostsze w realizacji, ale bardzo wolne. Szyfrowanie asymetryczne wymaga użycia techniki klucza prywatnego i publicznego, dzięki czemu działa znacznie szybciej, ale jest mniej bezpieczna. Niemniej jednak protokół SSL, z którego korzysta szyfrowanie asymetryczne jest obecnie wciąż uznawane za najbezpieczniejszą formę zabezpieczeń.