

UNIWERSYTET KAZIMIERZA WIELKIEGO w Bydgoszczy



Dokumentacja Techniczna Aplikacji Szyfrującej asynchronicznie RSA

*Bezpieczeństwo
systemów komputerowych*

2020/2021

Spis treści

1. RSA – Wstęp teoretyczny	3
2. Opis aplikacji	4
3. Kod aplikacji	7

1. RSA – Wstęp teoretyczny

Algorytm **Rivesta-Shamira-Adlemana (RSA)** – jeden z pierwszych i obecnie najpopularniejszych asymetrycznych algorytmów kryptograficznych z kluczem publicznym, zaprojektowany w 1977 przez Rona Rivesta, Adiego Shamira oraz Leonarda Adlemana. Pierwszy algorytm, który może być stosowany zarówno do szyfrowania, jak i do podpisów cyfrowych. Bezpieczeństwo szyfrowania opiera się na trudności faktoryzacji dużych liczb złożonych. Jego nazwa pochodzi od pierwszych liter nazwisk jego twórców.

Generowanie kluczy

W celu wygenerowania pary kluczy (prywatnego i publicznego) należy posłużyć się algorytmem:

- Wybieramy losowo dwie duże **liczby pierwsze** p i q (najlepiej w taki sposób, aby obie miały zbliżoną długość w bitach, ale jednocześnie były od siebie odległe wartościami – istnieją lepsze mechanizmy faktoryzacji, jeżeli liczba ma dzielnik o wartości bliskiej \sqrt{n}).
- Obliczamy wartość $n = pq$.
- Obliczamy wartość **funkcji Eulera** dla n : $\varphi(n) = (p - 1)(q - 1)$.
- Wybieramy liczbę e ($1 < e < \varphi(n)$) **względnie pierwszą** z $\varphi(n)$.
- Znajdujemy liczbę d , gdzie jej różnica z odwrotnością modularną liczby e jest podzielna przez $\varphi(n)$:

$$d \equiv e^{-1} \pmod{\varphi(n)}.$$

Ta liczba może być też prościej określona wzorem:

$$d \cdot e \equiv 1 \pmod{\varphi(n)}.$$

Klucz publiczny jest definiowany jako para liczb (n, e) , natomiast **kluczem prywatnym** jest para (n, d) .

Szyfrowanie i deszyfrowanie

Zanim zaszyfrujemy wiadomość, dzielimy ją na bloki m o wartości liczbowej nie większej niż n , a następnie każdy z bloków szyfrujemy według poniższego wzoru:

$$c \equiv m^e \pmod{n}.$$

Zaszyfrowana wiadomość będzie się składać z kolejnych bloków c . Tak stworzony **szyfrogram** przekształcamy na **tekst jawny**, odszyfrowując kolejne bloki c według wzoru:

$$m \equiv c^d \pmod{n}.$$

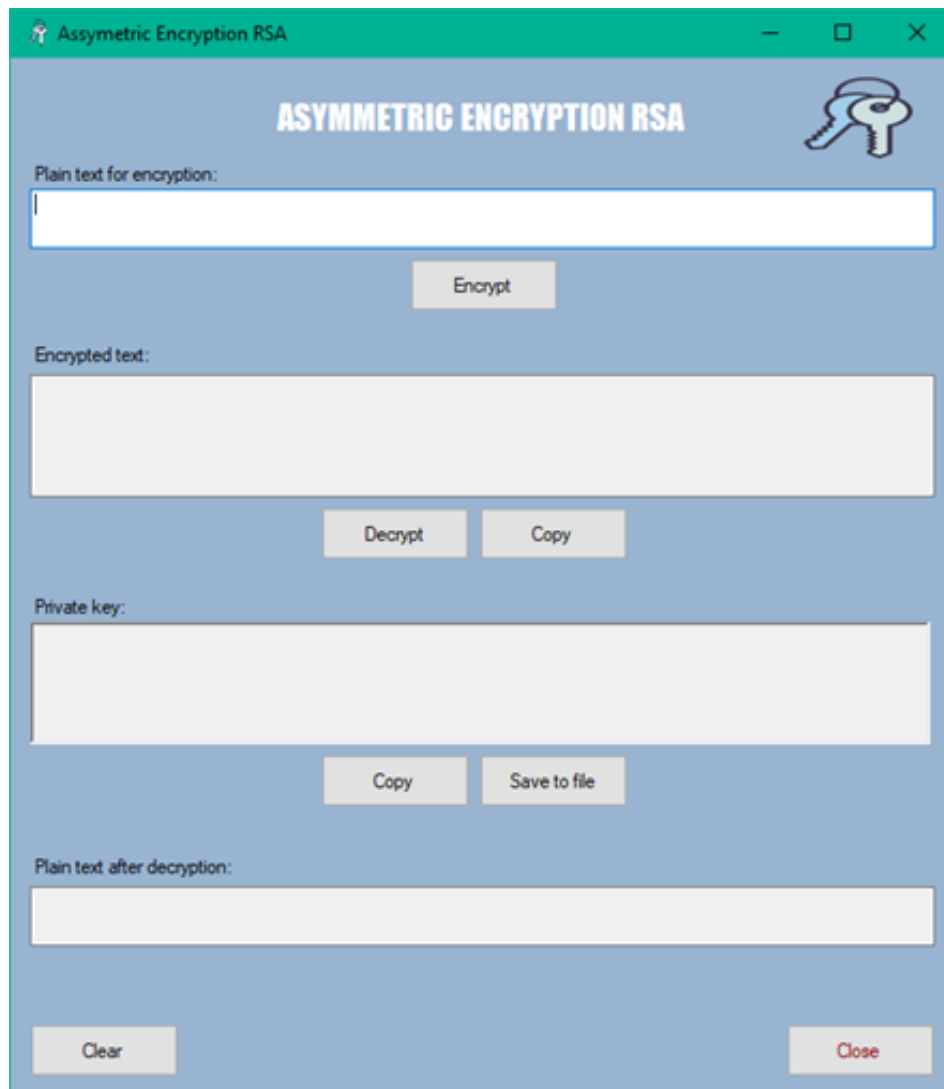
Własności operacji szyfrowania i deszyfrowania

Niech $C_{K_1}, D_{K_1}, C_{K_2}, D_{K_2}$ będą kolejno szyfrowaniem i deszyfrowaniem kluczami K_1 i K_2 . Wtedy zachodzi:

- $C_{K_1}(C_{K_2}(M)) = C_{K_2}(C_{K_1}(M))$ – przemienność operacji szyfrowania,
- $D_{K_1}(D_{K_2}(M)) = D_{K_2}(D_{K_1}(M))$ – przemienność operacji deszyfrowania.

Ze względów bezpieczeństwa nie powinno się stosować więcej niż 2 zagnieżdżone szyfrowania ze względu na ataki oparte na **chińskim twierdzeniu o resztach**.

2. Opis aplikacji




The screenshot shows a web application titled "Asymmetric Encryption RSA" in a browser window. The interface has a light blue background and a green header bar. The main content area is divided into several sections:

- Header:** The title "ASYMMETRIC ENCRYPTION RSA" is centered in bold white text. To the right is a logo of two interlocking keys.
- Plain text for encryption:** A text input field with a cursor at the start.
- Encrypt:** A button located below the first input field.
- Encrypted text:** A large, empty text area for the output of the encryption process.
- Decrypt:** A button located below the encrypted text area.
- Copy:** A button located to the right of the "Decrypt" button.
- Private key:** A large, empty text area for entering a private key.
- Copy:** A button located below the private key area.
- Save to file:** A button located to the right of the "Copy" button for the private key.
- Plain text after decryption:** A text input field for the output of the decryption process.
- Clear:** A button at the bottom left of the interface.
- Close:** A button at the bottom right of the interface.

Po uruchomieniu aplikacji wyświetli się interfejs z polami tekstowymi i przyciskami. Aby zaszyfrować dowolną wiadomość trzeba wpisać tekst w pierwszym od góry polu tekstowym, a następnie wcisnąć przycisk „Encrypt”.

Assymetric Encryption RSA

ASYMMETRIC ENCRYPTION RSA



Plain text for encryption:

Test message.

Encrypt

Encrypted text:

pvyS67wimW3l86voLfVMFdbLRWNi5u7CXCb2AYCNG46hJSInjGb2EQA8OZJR1RoL73ehpBLtq+9CvW4ECjW1+E+v0OpYMd4PYlA0YuZ/v/jYHO65ETtQhRCmkFnbU+E9o2aj3nRAW03vZvSalleVNvWtDYRywXP/6+E3leMYgCw=

DecryptCopy

Private key:

<RSAKeyValue><Modulus>p8sCvL54JEeRi9D5V3
+kLQgUkNnATeh+HO7j7vbh3yx07yQRqdV7Aj7dLzAknGuO3AZLPqUIKXkAlsDGkSIINIWBAYInwOxu+
2SG2zmKeow5mLjZqjKZlIP+7V6qEsx45dZy3RmaSYneZLbR/E79bGndZkob5exRcMM7/IY63PU=</Modulus>
<Exponent>AQAB</Exponent><P>
zE8990YU+lnkSoN6dQ3fILqCOF9HsQN37lpvMG6uG1Xxr/W2zHHkJanK14pTf1bXfeX+RbhD3iX6qQ4dJ8lgw==
</P>

CopySave to file

Plain text after decryption:

ClearClose

Następnie po wciśnięciu przycisku poniżej zostanie pokazana zaszyfrowana wiadomość oraz klucz prywatny. Oczywiście wyświetlone informacje można skopiować do schowka.

Dodatkowo kluz prywatny można zapisać do pliku tekstowego poprzez wciśnięci przycisku „Save to file”.

Assymetric Encryption RSA

ASYMMETRIC ENCRYPTION RSA

Plain text for encryption:

Encrypt

Encrypted text:

iWF03ZZMiDQEuDZIR3dl+/PfRsLdj+weOD5VqZnF9DsfVM/xyaCUcZzbsJJikorNa2WLTiCot0BleVxZjz+LSgL8/NjiT3H4hareiBIGv19vi+ioACrU25zegf1UDuBcyoDtKMSPRcsSvSaC+laK99VM0Ou5dXxRtLcPoWkV3I=

Decrypt Copy

Private key:

Copy Save to file

Plain text after decryption:

Other message.

Clear Close

Kolejną możliwością aplikacji jest deszyfrowanie wiadomości poprzez podanie zaszyfrowanego tekstu w drugim od góry polu tekstowym i wciśnięciu przycisku „Decrypt”. Następnie na samym dole pojawi nam się odszyfrowana wiadomość.

Dodatkowo na samym dole dostępne są jeszcze dwa przyciski:

- Clear – służy do wyczyszczenia wszystkich pól tekstowych;
- Close – służy do zamknięcia aplikacji.

3. Kod aplikacji

Funkcja przycisku, który szyfruje:

```
private void btnEncrypt_Click(object sender, EventArgs e)
{
    plainText = unicodeEncoding.GetBytes(tbPlainEncryption.Text); //text from first textbox (text to encrypt)

    //Encrypts data with RSA algorithm.
    encryptedText = Encryption(plainText, RSA.ExportParameters(false), false); //false for export the public key information
    tbEncrypted.Text = Convert.ToBase64String(encryptedText); //pass data to the second textbox (encrypted data)

    using (var RSA = new RSACryptoServiceProvider())
    {
        rTbKey.Text = RSA.ToXmlString(true); //pass private key to the third textbox
    }
}
```

Funkcja przycisku, który deszyfruje:

```
private void btnDecrypt_Click(object sender, EventArgs e)
{
    //Decrypts data with RSA algorithm.
    byte[] decryptedText = Decryption(encryptedText, RSA.ExportParameters(true), false); //true for export the private key information
    tbPlainDecryption.Text = unicodeEncoding.GetString(decryptedText); //pass data to the fourth textbox (decrypted data)

    using (var RSA = new RSACryptoServiceProvider())
    {
        string publicKey = RSA.ToXmlString(false); //pass public key to the variable
    }
}
```

Funkcja szyfrowania:

```
static public byte[] Encryption(byte[] Data, RSAParameters RSAKey, bool DoOAEPPadding)
{
    try
    {
        byte[] encryptedData;

        //New instance of RSACryptoServiceProvider to generate public and private key
        using (var RSA = new RSACryptoServiceProvider())
        {
            RSA.ImportParameters(RSAKey); //to include the public key information
            encryptedData = RSA.Encrypt(Data, DoOAEPPadding); //data for encrypt
        }
        return encryptedData;
    }
    catch (CryptographicException ex)
    {
        Console.WriteLine(ex.Message);
        return null;
    }
}
```

Funkcja deszyfrowania:

```
static public byte[] Decryption(byte[] Data, RSAParameters RSAKey, bool DoOAEPadding)
{
    try
    {
        byte[] decryptedData;

        //New instance of RSACryptoServiceProvider to generate public and private key
        using (var RSA = new RSACryptoServiceProvider())
        {
            RSA.ImportParameters(RSAKey); //to include the private key information
            decryptedData = RSA.Decrypt(Data, DoOAEPadding); //data for decrypt
        }
        return decryptedData;
    }
    catch (CryptographicException ex)
    {
        Console.WriteLine(ex.ToString());
        return null;
    }
}
```


4. Literatura

<https://docs.microsoft.com/pl-pl/dotnet/api/system.windows.forms.messagebox?view=net-5.0>

<https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.rsacryptoserviceprovider?view=net-5.0>

[https://pl.wikipedia.org/wiki/RSA_\(kryptografia\)](https://pl.wikipedia.org/wiki/RSA_(kryptografia))