

Лабораторна робота № 3 Асиметричне шифрування. Алгоритм RSA

Група КН-М922в

Автор Савичев О.В.

Мета

Дослідити і реалізувати механізм асиметричного алгоритму шифрування RSA.

Завдання

Розробити додаток обміну таємними посиланнями між двома клієнтами за допомогою алгоритму шифрування RSA

- Реалізувати алгоритм генерації ключів (public / private keys) для алгоритму RSA. Створити ключі заданої довжини (напр. 1024 біт)
- Реалізувати та продемонструвати роботу алгоритму шифрування та дешифрування повідомлення RSA
- Підтвердити роботу реалізованого алгоритму шляхом порівняння результату кодування з існуючим алгоритмом (наприклад, використовуючи утиліту openssl або вбудовані системи шифрування обраної мови програмування)

Хід роботи

RSA — криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.

RSA став першим алгоритмом такого типу, придатним і для шифрування, і для цифрового підпису. Алгоритм застосовується до великої кількості криптографічних застосунків.

Алгоритм RSA складається з 4 етапів: генерації ключів, шифрування, розшифрування та розповсюдження ключів.

Безпека алгоритму RSA побудована на принципі складності факторизації цілих чисел. Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари ключів (keypair). Відкритий ключ не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

Метод генерації ключів

```
public void GenerateKeys(out BigInteger N, out BigInteger publicE, out BigInteger privateD)
{
    // 1. Вибираються два великі прості числа p і q 512 біт завдовжки кожне
    BigInteger P;
    BigInteger Q;
    const int bits = 1024;
    const int bytes = bits / 8;
    do
    {
        P = Helper.Instance.GetPrime(bytes / 2);
        Q = Helper.Instance.GetPrime(bytes / 2);

        N = P * Q; // Обчислюється їх добуток N = P*Q - модуль
    } while (P == Q || N.ToArray().Length != bytes);
    Console.WriteLine($"P = {P}");
    Console.WriteLine($"Q = {Q}");
    Console.WriteLine($"N = {N}");

    // Обчислюється функція Ейлера phi(n) = (P-1)(Q-1)
    BigInteger PHI = (P - 1) * (Q - 1);

    // Вибирається ціле число E, таке, що 1 < E < PHI(n), E взаємно просте з phi(n)
    // Для підвищення швидкості шифрування відкритий показник E, вибирається невеликим,
    // звичайно 3, 17, 257 або 65537
    publicE = 65537;
    Console.WriteLine($"E = {publicE}");

    // За допомогою розширеного алгоритму Евкліда знаходиться число d, таке, що ed = 1 mod phi(n)
    var gcd = Helper.Instance.ExtendedEuclidean(publicE, PHI);
    var r = gcd.Item2 % PHI;
    privateD = r < 0 ? r + PHI : r;
    Console.WriteLine($"D = {privateD}");
}
```

Метод шифрування

```
public byte[] Encode(byte[] data, BigInteger publicE, BigInteger N)
{
    var blockSize = N.ToArray().Length;
    var msgAsBytes = _PaddingBlock(data, blockSize);
    var result = new byte[msgAsBytes.Length];
    var blocks = msgAsBytes.Length / blockSize;
    for (int i = 0; i < blocks; i++)
    {
        byte[] chunk = new byte[blockSize];
        Buffer.BlockCopy(msgAsBytes, i * blockSize, chunk, 0, blockSize);

        BigInteger chunkAsBigInteger = new BigInteger(chunk, true);
        var encodedChunk = BigInteger.ModPow(chunkAsBigInteger, publicE, N);
        var encodedBytes = encodedChunk.ToArray();
        Buffer.BlockCopy(encodedBytes, 0, result, i * blockSize, blockSize);
    }
    return result;
}
```

Метод дешифрування

```
public byte[] Decode(byte[] message, BigInteger privateD, BigInteger N)
{
    var blockSize = N.ToByteArray().Length;
    var decoded = new byte[message.Length];
    var blocks = message.Length / blockSize;
    for (int i = 0; i < blocks; i++)
    {
        byte[] chunk = new byte[blockSize];
        Buffer.BlockCopy(message, 0, chunk, i * blockSize, blockSize);

        BigInteger chunkAsBigInteger = new BigInteger(chunk, true);
        var decodedChunk = BigInteger.ModPow(chunkAsBigInteger, privateD, N);
        var decodedBytes = decodedChunk.ToByteArray();
        Buffer.BlockCopy(decodedBytes, 0, decoded, i * blockSize, blockSize);
    }

    if (decoded[0] == 0x00 && decoded[1] == 0x02) // Чи присутня сигнатура розширення
    {
        for (int i = 2; i < decoded.Length; i++)
        {
            if (decoded[i] == 0x00) // Пошук початку повідомлення
            {
                var size = decoded.Length - i - 1;
                var result = new byte[size];
                Buffer.BlockCopy(decoded, i + 1, result, 0, size);
                return result;
            }
        }
        return decoded;
    }
    else
    {
        return decoded;
    }
}
```

Результати роботи

Результати роботи моєї реалізації алгоритму RSA

```
D:\Univer\STBP\M922B_STBP\lab03\src\stbp_lab03_rsa\RSAServer\bin\Debug\net6.0\RSAServer.exe

Title: Асиметричне шифрування. Алгоритм RSA
Author: Савичев О.В.
Group: M922B

Генерація ключів...
P = 12686770061639804460615187831239700586021425512872051411364986385639176616864003983189003169095138584294586976961857
804085396307172317371464763357204651111
Q = 68843152768625591402353439261336062861320545971027892334762314137722173921914038550801878836639614454757669366836392
34783742733459976925097409649495520397
N = 87339724949389456934597608945789786859246555325103312988107923219544839988738180200415950913872089075217391035754230
649018692881852061652047173315422746900369661886474629759055729331857204197853764785219565241698551094044730048056756539
866417467903099966833808652431057488649644242547196371163071567769211067
E = 65537
D = 12737737324965506956084104342643984051767376837471008217347994722559921580364672266896190836241961447440801738250742
886359166067484657602121959847854046030283241445984999617622492308419932109175630879136809400747095406108082851835984114
162477532934775033926675014012121195340068491385059340968237613053662513

Пов?домлення [STR]: Savychv
Пов?домлення [HEX]: 5361767963686576

Пов?домлення п?сля доповнення (PKCS#v1.5) [HEX]:
0 1 2 3 4 5 6 7 8 9 a b c d e f
0000 00-02-E6-65-88-35-85-97-F0-D7-D9-E4-62-CA-0A-1C
0010 13-EA-C0-77-38-42-DA-58-CE-D0-F2-F9-7E-EC-EB-74
0020 5F-56-38-5B-C7-C7-8D-E8-E3-56-B7-0F-CC-1F-97-A4
0030 37-7B-DE-88-4F-C7-A0-C4-71-8E-95-B7-42-8A-19-05
0040 EB-65-86-0D-92-AE-53-4A-5E-D7-93-E0-89-CB-8B-AF
0050 3A-4F-C3-03-0E-63-26-96-B4-72-E8-39-32-D5-A6-59
0060 3E-B2-23-67-98-8A-12-87-28-95-2A-76-35-BE-3C-A4
0070 03-E5-7D-F2-C6-16-F6-00-53-61-76-79-63-68-65-76

Зашифроване пов?домлення [HEX]:
0 1 2 3 4 5 6 7 8 9 a b c d e f
0000 BB-C0-C1-A3-02-B3-B0-CB-6A-FE-EB-BB-25-60-07-21
0010 C7-98-AD-6E-44-F1-23-04-E9-59-D6-11-77-40-E0-8A
0020 32-E3-D3-A4-70-8A-CE-3E-24-3B-41-F5-2F-1A-41-DC
0030 40-EA-D5-78-53-9F-58-EB-EB-6C-94-94-00-15-DB-95
0040 A5-33-DC-70-B0-AC-FA-8C-5B-38-D2-0A-0E-15-12-7C
0050 6B-88-A7-21-01-60-4E-58-4D-B1-AC-C2-69-03-8A-0A
0060 56-1A-1D-02-A4-B0-C7-03-D4-75-63-D6-7F-EB-A0-5E
0070 72-8A-87-9D-00-8C-61-21-84-75-D7-11-64-AF-6C-67

Пов?домлення перед ф?льтруванням доповнення (PKCS#v1.5) [HEX]:
0 1 2 3 4 5 6 7 8 9 a b c d e f
0000 00-02-E6-65-88-35-85-97-F0-D7-D9-E4-62-CA-0A-1C
0010 13-EA-C0-77-38-42-DA-58-CE-D0-F2-F9-7E-EC-EB-74
0020 5F-56-38-5B-C7-C7-8D-E8-E3-56-B7-0F-CC-1F-97-A4
0030 37-7B-DE-88-4F-C7-A0-C4-71-8E-95-B7-42-8A-19-05
0040 EB-65-86-0D-92-AE-53-4A-5E-D7-93-E0-89-CB-8B-AF
0050 3A-4F-C3-03-0E-63-26-96-B4-72-E8-39-32-D5-A6-59
0060 3E-B2-23-67-98-8A-12-87-28-95-2A-76-35-BE-3C-A4
0070 03-E5-7D-F2-C6-16-F6-00-53-61-76-79-63-68-65-76

Розшифроване пов?домлення [HEX]: 5361767963686576
Розшифроване пов?домлення [STR]: Savychv
```

Результати роботи вбудованої системи шифрування алгоритму RSA

```
D:\Univer\STBP\M922B_STBP\lab03\src\stbp_lab03_rsa\RSAServer\bin\Debug\net6.0\RSAServer.exe

Робота вбудованої системи шифрування
MSG [STR]: Savychyev
MSG [HEX]: 53-61-76-79-63-68-65-76

KEY [PUB]:
  0 1 2 3 4 5 6 7 8 9 a b c d e f
0000 30-81-89-02-81-81-00-A3-36-36-C1-07-80-29-2F-12
0010 D8-80-23-44-62-E4-9C-25-2B-A1-1D-61-C5-C9-00-EF
0020 0B-71-C5-52-F1-EF-26-72-60-B8-E6-A1-35-3F-F0-F9
0030 32-99-1E-E2-F3-02-22-AE-EA-AF-F2-41-BE-C2-08-D6
0040 47-FF-A9-88-73-8F-76-FE-E0-A8-F9-57-48-47-2D-6D
0050 43-E8-0F-47-AD-3A-B4-1C-2E-1E-BA-1F-39-0E-22-5D
0060 11-7C-45-66-82-50-95-3D-6A-53-7B-B3-D3-7F-63-DA
0070 3C-49-7A-5F-CC-6C-CD-25-7F-AB-CA-EA-CC-17-19-BC
0080 02-7B-6B-F8-A0-76-49-02-03-01-00-01

KEY [PRV]:
  0 1 2 3 4 5 6 7 8 9 a b c d e f
0000 30-82-02-5D-02-01-00-02-81-81-00-A3-36-36-C1-07
0010 80-29-2F-12-D8-80-23-44-62-E4-9C-25-2B-A1-1D-61
0020 C5-C9-00-EF-0B-71-C5-52-F1-EF-26-72-60-B8-E6-A1
0030 35-3F-F0-F9-32-99-1E-E2-F3-02-22-AE-EA-AF-F2-41
0040 BE-C2-08-D6-47-FF-A9-88-73-8F-76-FE-E0-A8-F9-57
0050 48-47-2D-6D-43-E8-0F-47-AD-3A-B4-1C-2E-1E-BA-1F
0060 39-0E-22-5D-11-7C-45-66-82-50-95-3D-6A-53-7B-B3
0070 D3-7F-63-DA-3C-49-7A-5F-CC-6C-CD-25-7F-AB-CA-EA
0080 CC-17-19-BC-02-7B-6B-F8-A0-76-49-02-03-01-00-01
0090 02-81-80-0F-BD-DD-E0-16-4A-1D-2E-C6-21-8D-92-59
00a0 39-34-95-7E-DC-4B-68-DB-12-49-85-C6-88-93-0B-96
00b0 29-71-88-82-C5-A3-6F-D3-22-A0-AD-2C-4B-DE-0A-E5
00c0 96-CC-A0-ED-C0-03-02-59-A6-FD-6D-FE-C4-E8-FA-C0
00d0 07-6D-51-C4-65-19-32-39-E0-F7-65-C7-E6-2E-A5-DD
00e0 CF-04-E8-96-65-BA-30-DC-E7-A1-22-5C-65-59-20-F2
00f0 CA-65-D9-73-78-23-01-F2-4C-FC-74-A8-98-DA-89-86
0100 2F-33-4F-69-30-CA-59-BA-C8-7B-28-82-C2-FD-B9-D2
0110 BE-CD-45-02-41-00-D3-B7-09-71-1C-80-90-FC-4D-16
0120 17-90-05-B2-F7-18-CF-9B-70-EF-D2-37-4A-AF-3A-61
0130 ED-64-46-EB-01-2A-23-DE-04-42-54-24-8E-AA-68-51
0140 5C-2B-2B-2D-82-C6-07-4D-4C-9F-B3-96-EF-A2-72-56
0150 93-17-EA-85-77-93-02-41-00-C5-59-EA-67-42-77-A1
0160 D4-A7-01-33-AD-EF-86-3E-46-CD-CB-A3-70-58-BB-5F
0170 7F-0F-BE-5C-1E-F8-55-25-DF-1E-81-7A-07-77-22-B2
0180 EF-97-5B-FF-51-0A-11-05-DD-F3-23-1F-32-C8-A2-62
0190 83-E2-F1-81-D4-9F-8E-4C-33-02-41-00-BF-25-88-AE
01a0 C9-0F-02-D7-E8-39-43-F8-D2-4C-6C-6F-C8-31-0D-5C
01b0 59-2D-76-8E-92-65-8B-D1-77-7E-01-C5-2C-30-ED-23
01c0 7D-98-AB-FE-0A-C5-3A-33-F7-7D-D5-8E-39-55-7F-12
01d0 C9-30-43-17-25-A8-28-91-F5-39-9F-65-02-40-66-89
01e0 A0-4B-C4-72-2A-08-36-39-5E-A3-99-D6-F3-16-90-A8
01f0 A4-A3-A3-C6-BF-08-62-A2-B2-74-78-EC-AC-BF-AF-B6
0200 F0-33-5E-C8-0D-96-84-DB-0C-DE-0B-4F-EA-EF-75-FB
0210 A2-A3-1D-31-99-E4-12-8F-0E-B8-ED-A9-DE-97-02-41
0220 00-A2-A2-DE-B6-8C-EA-5A-BE-27-7E-FD-E1-AD-66-46
0230 BA-9C-81-BF-51-41-23-85-BD-DB-1D-66-81-E9-34-73
0240 C8-B9-D1-25-A3-92-42-13-08-69-68-06-8D-7F-5C-89
0250 3C-73-67-06-11-B5-E0-CD-72-46-BB-5D-84-1C-BF-C6
0260 6B

MSG [ENC]:
  0 1 2 3 4 5 6 7 8 9 a b c d e f
0000 86-55-17-F0-39-48-E8-11-D7-86-52-8C-EC-9D-60-71
0010 BD-0C-6C-2A-0E-DD-1C-EF-59-92-B9-9A-33-2B-5A-5F
0020 4E-04-75-A4-B7-C0-0B-E9-BD-6D-4B-B1-67-68-04-F9
0030 B7-CE-E6-92-C6-17-62-EE-5D-7C-32-16-0C-9D-80-EA
0040 AA-EB-08-3F-2C-EF-11-1D-30-D3-A0-9C-85-46-83-36
0050 26-A5-56-A3-0A-4B-75-5B-EE-BE-87-9A-09-B1-A3-2C
0060 7E-86-38-4A-30-2E-D2-F4-42-09-83-7A-16-E6-F7-2F
0070 98-03-36-81-D5-76-E9-1D-FB-01-AF-1C-3F-CE-F5-82

MSG [DEC]: 5361767963686576
MSG [STR]: Savychyev
```

Висновок

Дослідив і реалізував механізм асиметричного алгоритму шифрування RSA.