Using the RC4 encryption algorithm at the server

ZHAO Chun yang
JiLin Business And Technology College,changchun ,china,130062
e-mail address:804543056@qq.com

Abstract: The security of data has become a popular topic in computer science，however,encrypted data is the only way to solve. Applying various kind of encrypted algorithms into program is very important. One of the simply and security algorithms is the RC4, it was created by Ronald Rivest of RSA Security in 1987. It is one of the most widely-used software stream cipher and is used in popular protocols such as Secure Sockets Layer (SSL) (to protect Internet traffic) and WEP (to secure wireless networks). RC4 is used in many commercial software packages such as Lotus Notes and Oracle Secure SQL. This paper mainly introduces the application of c # RC4 encryption method on the server side

Keywords: RC4，c#，class，server，encryption, decrypt

RC4 加密算法在服务器端的应用

赵春阳
吉林工商学院，长春市，中国，130062
E-mail：804543056@qq.com

【摘要】数据安全是计算机科学中的热门话题，但是加密算法才是解决它的唯一方法。在程序中应用不同的加密算法是很重要的，RC4 加密算法就是一种既简单又安全的算法，是 RSA 中的 Ronald Rivest 在 1987 年设计的，它是使用最广泛的软件流密码之一，并且广泛应用在 SSL 和 WEP 层协议中。RC4 被用在许多诸如 Lotus Notes 和 Oracle Secure SQL 商业软件包中。本文着重介绍用 c#语言编写的 RC4 加密方法在服务器端的使用。

【关键词】RC4；c#；类；服务器；加密；解密

1、RC4 算法简介及算法说明

RC4 是目前密钥长度达到 128 位的最安全的加密算法之一。RC4 算法的原理很简单，包括初始化算法和伪随机子密码生成算法两大部分。假设 S-box（长度是任意的，通常是 256 字节）长度和加密密钥长度均为 n。先来看看算法的初始化部分（用类 C 伪代码表示）

```
for (i=0; i<n; i++)
 s[i]=i;
j=0;
for (i=0; i<n; i++)
{
      j=(j+s[i]+k[i])%256;
      swap(s[i], s[j]);
}
```

Swap 函数实现两个数据的交换功能，如：a=5，b=6，执行函数 swap(a, b)，运算后的结果就变成，a=6, b=5 了。

在初始化的过程中，密钥的主要功能是将 S-box 搅乱，i 确保 S-box 的每个元素都得到处理，j 保证 S-box 的搅乱是随机的。而不同的 S-box 在经过伪随机子密码生成算法的处理后可以得到不同的子密钥序列，并且，该序列是随机的：

```
i=0；j=0;
while (明文未结束)
{           ++i%=n;
            j=(j+s[i])%n;
            swap(s[i], s[j]);
            sub_k=s((s[i]+s[j])%n);
}
```

得到的子密码 sub_k 用以和明文进行异或（xor）运算，得到密文，解密过程也完全相同。

978-1-935068-09-9

2、C#语言描述服务器端 RC4 算法

Rc4 算法在很多语言中都有描述，本文主要使用 c# 语言在服务器端实现加密和解密操作。首先要建立一个基类 Encryption_toBase，主要是为多种加密算法（RC4、DES、RSA 三种算法）提供一个统一接口，还需要一个十六进制字符串转换方法来进行编码解码。

引用空间：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
```

Encryption_toBase.cs 文件代码：

```
using Systemode;
using Systemode.Collections.Generic;
using Systemode.Text;
namespace Encryption_C_Sharp
{/// <summary>
  /// 用于加密的基类
  /// </summary>
  public class Encryption_toBase
  { /// <summary>
   /// 用于字节码和字符串之间转换的编码转换器
   /// </summary>
   static public Encoding Char_coding = Encoding.Default;
  public enum Char_codingrMode { Base64Char_codingr, HexChar_codingr };
  /// <summary>
  /// 加密带编码模式的字符串
  /// </summary>
  /// <param name="data">加密的数据</param>
  /// <param name="password">密码</param>
  /// <param name="emode">编码模式</param>
  /// <returns>加密后经过编码的字符串</returns>
  public String Encrypt(String data, String password, Encryption_toBase.Char_codingrMode emode)
  {   if (data == null || password == null) return null;
      if (emode == Char_codingrMode.Base64Char_codingr)
          return Convert.ToBase64String(Encrypt_Ex(Char_coding.GetBytes(data), password));
      else
          return ByteToHex(Encrypt_Ex(Char_coding.GetBytes(data), password));  }
  /// <summary>
  ///解密带编码模式的字符串
  /// </summary>
  /// <param name="data">解密的数据</param>
  /// <param name="password">密码</param>
  /// <param name="emode">编码模式</param>
  /// <returns>明文</returns>
  public String Decrypt(String data, String password, Encryption_toBase.Char_codingrMode emode)
  {  if (data == null || password == null) return null;
if(emode==Char_codingrMode.Base64Char_codingr)
          return Char_coding.GetString(Decrypt_Ex(Convert.FromBase64String(data), password));
      else
          return Char_coding.GetString(Decrypt_Ex(HexToByte(data), password));  }
  /// <summary>
  /// 加密
  /// </summary>
  /// <param name="data">要加密的数据</param>
  /// <param name="password">密码</param>
  /// <returns>加密后经过默认编码的字符串</returns>
  public String Encrypt(String data, String password)
  { return Encrypt(data, password, Char_codingrMode.Base64Char_codingr);
  }
  /// <summary>
  /// 解密
  /// </summary>
  /// <param name="data">要解密的经过编码的数
```

```
据</param>
        /// <param name="password">密码</param>
        /// <returns>明文</returns>
        public String Decrypt(String data, String password)
        {       return Decrypt(data, password, Char_codingrMode.Base64Char_codingr);
                }
          /// <summary>
          /// 加密
          /// </summary>
          /// <param name="data">要加密的数据</param>
          /// <param name="password">密钥</param>
          /// <returns>密文</returns>
          virtual public Byte[] Encrypt_Ex(Byte[] data, String password) { return null; }
          /// <summary>
          /// 解密
          /// </summary>
          /// <param name="data">要解密的数据</param>
          /// <param name="password">密码</param>
          /// <returns>明文</returns>
          virtual public Byte[] Decrypt_Ex(Byte[] data, String password) { return null; }
          static public Byte[] HexToByte(String strHex)
          {    // 两个字节代表一个十六进制
               Int32 strHex_Len = strHex.Length;
               if (strHex_Len <= 0 || 0 != strHex_Len % 2)
               {       return null;          }
               Int32 Byte_Count = strHex_Len / 2;
                UInt32 temp1, temp2;
               Byte[]       Buffer_Array       =       new Byte[Byte_Count];
        for (Int32 i = 0; i < Byte_Count; i++)
        { temp1 = (UInt32)strHex[i * 2] - (((UInt32)strHex[i * 2] >= (UInt32)'A') ? (UInt32)'A' - 10 : (UInt32)'0');
          if (temp1 >= 16) return null;
          temp2 = (UInt32)strHex[i * 2 + 1] - (((UInt32)strHex[i * 2 + 1] >= (UInt32)'A') ? (UInt32)'A' - 10 : (UInt32)'0');
          if (temp2 >= 16) return null;
          Buffer_Array[i] = (Byte)(temp1 * 16 + temp2);
        }
        return Buffer_Array;
  }
      static public String ByteToHex(Byte[] Byte_Array)
      { if (Byte_Array == null || Byte_Array.Length < 1) return null;
          StringBuilder Str_Byte = new String-Builder(Byte_Array.Length * 2);
          for (int i = 0; i < Byte_Array.Length; i++)
        { if ((UInt32)Byte_Array[i] < 0) return null;
          UInt32 k = (UInt32)Byte_Array[i] / 16;
          Str_Byte.Append((Char)(k + ((k > 9) ? 'A' - 10 : '0')));
          k = (UInt32)Byte_Array[i] % 16;
          Str_Byte.Append((Char)(k + ((k > 9) ? 'A' - 10 : '0')));
        }
          return Str_Byte.ToString();
      }
    }
```

接口代码

```
  interface EncryptionTo Decrypt
  {   /// <summary>
      /// 加密
      /// </summary>
      /// <param name="data">要加密的数据</param>
      /// <param name="password">密钥</param>
      /// <returns>密文</returns>
      Byte[] Encrypt_Ex(Byte[] data, String password);
       /// <summary>
       /// 解密
       /// </summary>
       /// <param name="data">要解密的数据</param>
                /// <param name="password">密码</param>
                 /// <returns>明文</returns>
                Byte[] Decrypt_Ex(Byte[] data, String password);
          }
  }
```

RC4.cs 文件代码：

```
  using Systemode;
```


978-1-935068-09-9 © 2010 SciRes.

```
using Systemode.Collections.Generic;
using Systemode.Text;
namespace Encryption_C_Sharp
{   public class RC4_cSharp : Encryption_toBase
    { static public RC4_cSharp RC4 = new RC4_cSharp();
```

加密代码段

```
    public override Byte[] Encrypt_Ex(Byte[] data, String password)
    {   if (data == null || password == null) return null;
        Byte[] Out_Data = new Byte[data.Length];
        Int64 i = 0;
        Int64 j = 0;
        Byte[] Packaging_Box = GetKey(Char_coding.GetBytes(password), 256);
        // 加密
        for (Int64 K = 0; K < data.Length; K++)
        {i = (i + 1) % Packaging_Box.Length;
         j = (j + Packaging_Box[i]) % Packaging_Box.Length;
         Byte Temp_Byte = Packaging_Box[i];
         Packaging_Box[i] = Packaging_Box[j];
         Packaging_Box[j] = Temp_Byte;
         Byte a = data[K];
         Byte b = Packaging_Box[(Packaging_Box[i] + Packaging_Box[j]) % Packaging_Box.Length];
         Out_Data[K] = (Byte)((Int32)a ^ (Int32)b);
        }
        return Out_Data;
    }
```

解码程序

```
    public override Byte[] Decrypt_Ex(Byte[] data, String password)
    { return Encrypt_Ex(data, password); }
    /// <summary>
    /// 打乱密码
    /// </summary>
    /// <param name="password">密码</param>
    /// <param name="kLen">密码箱长度</param>
    /// <returns>打乱后的密码</returns>
    static private Byte[] GetKey(Byte[] password, Int32 kLen)
    {Byte[] Packaging_Box = new Byte[kLen];
     for (Int64 i = 0; i < kLen; i++)
     { Packaging_Box[i] = (Byte)i; }
       Int64 j = 0;
       for (Int64 i = 0; i < kLen; i++)
        {j = (j + Packaging_Box[i] + password[i % password.Length]) % kLen;
         Byte Temp_Byte = Packaging_Box[i];
         Packaging_Box[i] = Packaging_Box[j];
         Packaging_Box[j] = Temp_Byte;
        }
       return Packaging_Box;
     }
   }
}
```

由于 RC4 算法加密是采用的 xor，所以，一旦子密钥序列出现了重复，密文就有可能被破解，不过对于密钥长度达到 128 位的，RC4 还是非常安全的。

致　谢

感谢省教育厅科研处对客户端/服务器端密码安全设置项目的大力支持，也感谢课题组成员对改课题的鼎力相助。

References (参考文献)

[1] HUANG Dao-lin; YANG Jun FPGA implementation of RC4 encrypt algorithm, Journal of Yunnan University(Natural Sciences Edition),,2009-12-21
黄道林; 杨军,RC4 加密算法的 FPGA 设计与实现, 云南大学学报, ,2009-12-21

[2] XU KeJian The analysis of the number of fixed points in the key extending algorithm of RC4 Science in China(Series A:Mathematics) , 2008-05-26,

[3] XU Tian-bing, Comparison Between Class of C# And C++ Computer Knowledge and Technology(Academic Exchange) 2007-09-11
许天兵,C#类与 C++类的比较, 电脑知识与技术(学术交流),2007-09-11

[4] DU Yu-lan ZHAO Lei Hash Algorithms Research Based on C# Language Network & Computer Security 2007- 08-11
杜玉兰; 赵磊, 基于C#的HASH算法探析, 计算机安全,2004-08-11.