.NET Framework 对称算法类分析及其通用实现①

胡 振

(南充职业技术学院 信息与管理工程系, 南充 637000)

摘 要: 在.NET Framework 平台开发软件系统时, 一般采用某种特定算法实现敏感信息的加密, 这影响了软件的扩展性和维护性. 为此, 研究了.NET Framework 对称算法类的继承层次结构, 分析了其加密与解密的实现流程, 提出了通用对称算法加密实现类的设想, 并进行了可行性分析. 然后介绍了通用对称算法加密类的构成, 阐明了各成员的设计思路, 给出了用 VB.NET 实现的完整代码. 最后以应用实例表明, 该类使用简单方便, 具有良好的扩展性和维护性, 其设计思想可推广到其它加密算法类.

关键词: .NET Framework; 对称算法; 通用; 加密; 类

Analysis of .NET Framework Symmetric Algorithm and Its General Implementation

HU Zhen

(Information & Management Engineering, Nanchong Professional Technic College, Nanchong 637000, China)

Abstract: When the .NET Framework platform developed software systems,it generally used a particular algorithm to achieve the encryption of sensitive information.This affects the scalability and maintainability of software.To this end,studied the inheritance hierarchy for the .NET Framework symmetric algorithm class,analysed its process of the encryption and decryption, proposed a assumption of general symmetric algorithm encryption class,and made a feasibility analysis.Then introduced the composition of the general symmetric algorithm encryption class, explained the design ideas of the various members,and gave the complete code achieved through VB.NET.Finally used an example to show that this class is very convenient and its designing ideas can be extended to other encryption algorithms.

Key words: .NET Framework; symmetric algorithm; general; encrypt; class

在软件系统设计中, 数据安全性是我们考虑的重中之重, 特别是像信息系统的用户信息、银行系统的账户信息之类的敏感数据都需要进行加密处理[1].

.NET 平台开发人员都知道, 使用.NET Framework 提供的加密算法类进行信息的加解密是非常简单易行的, 而且各种加密算法的实现方案早已屡见不鲜. 其实, 在.NET Framework 中有些加密算法(比如同属于对称密码体制[2]的 DES、RC2、Rijndael 和 TripleDES 等算法)的实现过程是完全相同的, 若需在不同的软件系统中采用其中的不同算法进行加密, 则会多次重复编写几乎完全相同的实现过程, 这既不符合 OOP 设计思想, 也不利于软件的维护.

那么, 我们能否改变思路, 设计一个可扩展、易维护的通用加密算法实现类, 以此实现相同类型的各种算法加解密过程呢? 在详细研究了.NET Framework 的加密算法类继承层次结构与实现流程之后, 我们能够得到肯定的回答. 限于篇幅, 本文仅讨论对称算法类的通用实现.

1 .NET Framework对称算法类分析

1.1 对称算法类的继承层次结构

在 Microsoft 的.NET Framework 的开发平台中, System.Security.Cryptography 命名空间提供加密服务, 包括安全的数据编码和解码以及许多其它操作, 例如

① 收稿时间:2012-05-28;收到修改稿时间:2012-06-28

散列法、随机数字生成和消息身份验证. .NET Framework 通过提供者模式实现加密算法的动态扩展, SymmetricAlgorithm 类、AsymmetricAlgorithm 类和 Hash 类分别为对称算法、非对称算法和 Hash 算法的基类, 然后分别动态扩展出不同算法, 最后是每种算法的具体实现, 它们在命名空间中被组织为三层继承结构. 其中对称算法类的继承层次结构如图 1 所示.

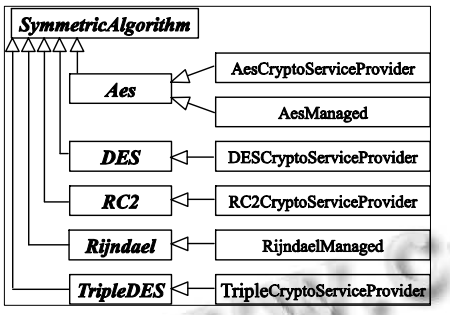

图 1　对称算法类的继承层次结构

1.2 称算法类的加解密实现流程

在.NET Framework 中实际应用对称算法进行加解密时, 一般是基于上述第三层的特定算法实现类来进行的. 其实现流程[3]为:

① 创建特定算法实现类的实例;

② 导入密钥和初始化向量→设置 Key 属性和 IV 属性;

③ 创建对称加(解)密器对象→调用 CreatencryptorCreateDecryptor)方法;

④ 定义加(解)密后的输出流→创建 MemorytStream 类的实例;

⑤ 将输出流链接到对称加(解)密器→创建 Cryptotream 类的实例;

⑥ 将明(密)文编码为字节序列→调用 Encoding 类或 Convert 类对象的相应方法;

⑦ 将明文字节序列加(解)密→调用 Cryptotream 对象的 Write 方法;

⑧ 将加(解)密后的输出流转换为字符串→调用 Convert 类或 Encoding 对象的相应方法.

1.3 通用对称算法加密类的可行性分析

我们的设想是: 对应于对称算法类的第一层设计并实现一个能够支持全部五种算法的通用加密实现类, 以便在应用时使用其中任一算法进行加解密. 其原理如图 2 所示.

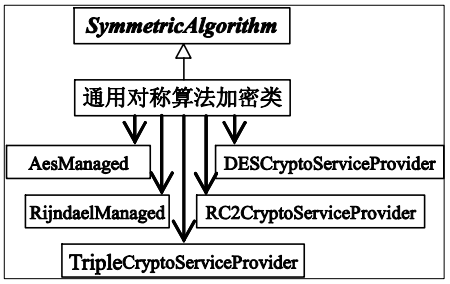

图 2　通用对称加密算法类原理

分析前述对称算法类的加解密实现流程可知, 步骤②、③、⑤、⑦涉及到对称算法实现类的一些成员操作. 经研究.NET Framework 文档发现, 这些成员都从第一层继承到了第二层和第三层, 这是能够针对第一层设计通用对称算法加密类的关键. 在此基础上仅需解决两个问题:

其一, 步骤①要求创建特定对称算法实现类的实例, 而第一层的 SymmetricAlgorithm 类是抽象类, 不能直接实例化.

解决之道是利用 Create(String)方法, 该方法接受 SymmetricAlgorithm 类的特定实现名为参数, 返回一个用于执行对称算法的指定加密对象.

其二, 各种对称算法所支持的密钥和初始化向量长度不同.

默认情况下, .NET Framework 对称算法采用密码块链接(CBC)模式进行加密和解密, 需要使用初始化向量来加密第一个数据块. 由其运算机制可知, 初始化向量的长度必须与数据块的长度相同.

访问特定算法类的 LegalKeySizes 属性和 LegallockSizes 属性, 可分别获知其支持的密钥长度[4]和数据块长度; 而查询 KeySize 属性和 BlockSize 属性, 则能分别得到默认的密钥长度和数据块长度. 由此可以发现, .NET Framework 已将各种对称算法的默认密钥长度设置为最大, 而数据块长度则仅 Rijndael 算法可以 64bit 为增量在 128bit~256bit 之间改变, 其余皆为固定值. 所以, 可读取 KeySize 属性和 BlockSize 属性来分别确定特定对称算法的密钥长度和初始化向量长度.

2 通用对称算法加密类设计与实现

2.1 用对称算法加密类的设计

将通用对称算法加密类命名为 allSymmAlgo, 它包括构造函数和四个方法.

2.1.1 构造函数

功能: 以特定算法初始化本类的实例.

参数: 对称算法名. 在实际应用时, 其参数值只能取 “AES” 、 “DES” 、 “RC2” 、 “RIJNDAEL” 和 “TRIPLEDES”(或“3DES”)之一; 无参构造函数用于设置默认算法.

2.1.2 CreateCSP 方法

功能: 创建特定算法的 CSP(或 Managed)对象.

虽然.NET Framework 的抽象类无法实例化, 但却可以特定对称算法名为参数调用其 Create(String)方法, 在该层次直接创建相应算法实现类的 CSP(或 Managed)对象. 在实际应用时, 可将其作为实参传送给 GenerateKeyIV、Encrypt 和 Decrypt 方法.

2.1.3 GenerateKeyIV 方法

功能: 根据密码字符串生成特定算法所需的密钥和初始化向量.

参数: 特定算法 CSP、密码.

安全的密钥和初始化向量应为动态生成的随机值, 可利用 Rfc2898DeriveBytes 类的 GetBytes 方法实现之. 将密码、salt 值和迭代次数传送给构造函数, 创建 Rfc2898DeriveBytes 类的一个实例, 即可调用其 GetBytes 方法生成密钥和初始化向量. 该方法仅需一个参数——要生成的伪随机密钥字节数, 其值可由特定算法类的 KeySize 属性和 BlockSize 属性值确定. 在实际应用时, 可将生成的密钥和初始化向量作为实参传送给 Encrypt 方法和 Decrypt 方法.

2.1.4 Encrypt 方法和 Decrypt 方法

功能: 以特定算法、给定的密钥和初始化向量加密明文和解密密文.

参数: 特定算法 CSP、明文(密文)、密钥、初始化向量.

在.NET Framework 中使用对称算法进行加密和解密时, 需要为特定算法实现类的 CSP(或 Managed)对象设置几个相关属性, 主要包括:

① 密钥和初始化向量: 虽然.NET Framework 在实例化特定算法实现类时会自动将它们设置为随机值, 但由于对称算法在加密和解密时必须使用相同的密钥和初始化向量, 因此应按特定算法的要求直接给出或设计专门的方法来生成这两个值. 在本类中用 GenerateKeyIV 方法来生成动态的密钥和初始化向量.

② 运算模式: .NET Framework 对称算法支持五种运算模式, 可查询 CipherMode 枚举. 默认为 CBC 模式.

③ 填充模式: .NET Framework 对称算法支持五种填充模式, 可查询 PaddingMode 枚举. 默认为 PKCS7 模式.

由于在.NET Framework 中初始化特定算法类的新实例时, 默认属性总是被设置为尽可能地安全可靠, 因此在本类的 Encrypt 方法和 Decrypt 方法的签名中没有包括 CipherMode 和 PaddingMode 参数, 这样可简化其设计与调用.

在实际应用时, GenerateKeyIV、Encrypt 和 Decrypt 方法的功能实现是对应于特定对称算法的, 因此在它们的签名中均包括一个 SymmetricAlgorithm 类型的参数, 其值可由 CreateCSP 方法返回.

设计完成之后, 通用对称算法加密类的成员构成如图 3 所示.

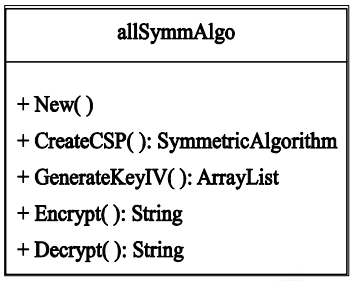

图 3 通用对称算法加密类的成员构成

2.2 通用对称算法加密类的实现

我们采用 Visual Basic 2010(基于.NET Framework 4.0 开发平台)[5,6], 实现了通用对称算法加密类. 为节省篇幅, 下面的代码省略了导入命名空间的语句和解密函数过程.

```
Public Class allSymmAlgo
' 变量 AlgoName 存放对称算法名
Private AlgoName As String
''' 无参构造函数. 默认 AES 算法
Public Sub New()
  AlgoName = "AES"
End Sub
''' 有参构造函数. 指定特定算法
Public Sub New(ByVal myAlgo As String)
  AlgoName = UCase(Trim(myAlgo))
End Sub
```

```
''' 创建特定对称算法的 CSP(或 Managed)对象
Public Function CreateCSP() As SymmetricAlgorithm
Return SymmetricAlgorithm.Create(AlgoName)
End Function
''' 生成密钥和初始化向量
' 参数: 特定算法 CSP、密码; 返回: 密钥、初始化向量
Public Function GenerateKeyIV(ByVal algoCSP As SymmetricAlgorithm, ByVal pwd As String) As ArrayList
' 以强随机值填充得到 8 字节 salt 值
Dim salt(7) As Byte
Dim rngCSP As New RNGCryptoServiceProvider()
rngCSP.GetBytes(salt)
' 产生特定对称算法的密钥和初始化向量. 存入 ArrayList 中
Dim rfc As New Rfc2898DeriveBytes(pwd, salt)
Dim KeyIv As New ArrayList
KeyIv.Add(rfc.GetBytes(algoCSP.KeySize / 8))
KeyIv.Add(rfc.GetBytes(algoCSP.BlockSize / 8))
Return KeyIv
End Function
''' 加密
' 参数: 特定算法 CSP、明文、密钥、初始化向量; 返回: 密文
Public Function Encrypt(ByVal algoCSP As SymmetricAlgorithm, ByVal plainText As String, ByVal KeyIv As ArrayList) As String
' 将明文字符串转换为字节数组
Dim data() As Byte = UTF8Encoding. UTF8. GetBytes(plainText)
' 导入密钥和初始化向量
algoCSP.Key = KeyIv.Item(0)
algoCSP.IV = KeyIv.Item(1)
' 创建加密器对象
Dim ICT As ICryptoTransform = algoCSP. Create Encryptor()
' 定义加密后的输出流
Dim ms As New MemoryStream()
' 定义将目标流链接到加密转换器的流
Using cs As New CryptoStream(ms, ICT, CryptoStreamMode.Write)
' 将明文字节序列写入当前 CryptoStream
cs.Write(data, 0, data.Length)
End Using
ms.Close()
' 返回密文字符串
Return Convert.ToBase64String(ms.ToArray())
End Function
End Class
```

3 通用对称算法加密类的应用

在应用对称算法加密类时, 先以对称算法名为参数(无参则默认 AES 算法)初始化该类的实例, 再用给定的密码调用 GenerateKeyIV 方法生成密钥和初始化向量, 其后即可分别调用 Encryp 方法和 Decrypt 方法进行信息的加密和解密.

下面以 TripleDES 算法创建通用对称算法加密类的实例, 用给定的密码生成密钥和初始化向量, 然后对字符串加密和解密. 其运行结果如图 4 所示.

```
' 定义明文和密码
Dim plainText As String = "对称算法加密"
Dim pwd As String = "By Huzhen 2012.05"
' 以 3des 算法创建通用对称算法加密类的实例
Dim allSA As New allSymmAlgo("3des")
' 生成动态密钥和初始化向量
Dim KeyIV As ArrayList = allSA.Generate KeyIV(allSA.CreateCSP, pwd)
' 分别加密和解密指定的字符串
Dim cipherStr As String = allSA.Encrypt (allSA.CreateCSP, plainText, KeyIV)
Dim plainStr As String = allSA.Decrypt (allSA. CreateCSP, cipherStr, KeyIV)
' 用消息框显示结果
MsgBox("原文: " & plainText & vbCrLf & "加密: " & cipherStr & vbCrLf & "解密: " & plainStr)
```

4 结语

.NET Framework 的 System.Security.Cryptography 命名空间包括了丰富的加密算法类, 它们按继承关系组成三层结构. 本文分析了对称算法类的继承层次结

构及其加解密实现流程，在此基础上提出了通用对称算法加密实现类的构想. 在进行了可行性分析之后，详细介绍了该类的成员构成、设计思路及 VB.NET 代码，并在最后给出了一个应用实例.

图 4 实例运行结果

实践证明，应用该类实现信息的加解密非常简便易行. 而且，即使.NET Framework 的未来版本加入新的对称算法，该类的代码和相应的软件系统也无需进行任何修改，因此具有良好的扩展性和维护性. 这种设计思想同样适用于非对称算法类和 Hash 算法类.

参考文献

1 吕君可.基于 AES 与 HASH 的软件数据保护.计算机系统应用,2011,20(3):210–213.

2 Stinson DR. Cryptography Theory and Practice.冯登国,等译. 第 3 版.北京:电子工业出版社,2009.126–138.

3 秦艳琳,吴晓平.基于.NET 的 TripleDES 算法在网络传输中的实现.计算机应用与软件,2010,27(11):53–55.

4 Petroutsos E, Mansfield R. Visual Basic.NET Power Tools.高春蓉,朱军,夏永存,等译.北京:电子工业出版社,2004. 132–133.

5 Sheldon B, Hollis B, Sharkey K. Professional Visual Basic 2010 and .NET 4. 彭珲,余科洋译.北京:清华大学出版社, 2011. 267–279.

6 王涛.你必须知道的.NET.第 2 版.北京:电子工业出版社, 2011. 224–230.

(上接第 115 页)

detect application. IEE Proceedings- Radar, Sonar and Navigation(S1350-2395), 2004,151(6):351–357.

9 Hlinomaz P, Hong L. A multi-rate multiple model track-before -detect particle filter. Mathematical and computer-modeling (S0895-7177), 2009,49:146–162.

10 龚亚信,杨宏文,胡卫东,等.基于多模粒子滤波的机动弱目标检测前跟踪.电子与信息学报,2008,30(4):941–944.

11 高山,毕笃彦,魏娜.处于一种基于 UPF 的小目标 TBD 算法.第十四届全国图象图形学术会议,福州,2008:265–271.

12 刘彬.微弱目标检测前跟踪算法研究[硕士学位论文].成都:电子科技大学,2010.

13 Barshalom Y, Fortmann TE. Tracking and Data Association. New York: Academic Press, 1988. 123–272.

14 Reed I, Gagliardi R, Stotts L. Optical moving target detection with 32D matched filtering. IEEE Trans. on Aerospace and Electronic Systems, 1988,24(4):327–336.

15 Kramer JD, Reid WS. Track before detect processing for a range 2 ambiguous radar. IEEE National Radar Conference. New York: IEEE, 1993.