高级加密标准算法 Rijndael 的分析与应用 *

张青凤[1]，张凤琴[2]，王 蓉[2]
（1.运城学院，公共计算机教学部，山西 运城 044000；
2.空军工程大学，计算机系，陕西 西安 710077）

摘 要：介绍了高级加密标准 Rijndael 算法的背景和特点，深入研究和分析了该算法的实现原理和过程，并给出了该算法在文件加解密过程实现的关键代码。

关键词：Rijndael 算法；状态；密钥
中图分类号：TP301.6 **文献标识码**：A **文章编号**：1674-7720(2012)07-0068-03

Analysis and application of the algorithm on advanced encryption standard Rijndael

Zhang Qingfeng[1], Zhang Fengqin[2], Wang Rong[2]
(1.Department of Public Computer Science of Yuncheng University, Yuncheng 044000, China;
2.Department of Public Computer Science of Air force Engineering University, Xi'an 710077, China)

Abstract: This paper briefly introduces the background and characteristics of algorithm of advanced encryption standard Rijndael, the in-depth study of the algorithm is analyzed and the realization of the principle and the process, the algorithm are given in the document encryption process the essential code.

Key words: Rijndael algorithm; state; key

1 Rijndael 算法的背景

AES(Advanced Encryption Standard)是美国联邦标准局于 1997 年开始向全世界征集的加密标准[1]，属于对称加密算法，代表了当今最先进的编码技术，最终获胜的是 RijnDael 算法。统计显示，即使使用目前世界上运算速度最快的计算机，穷尽 128 bit 密钥也需要几十亿年的时间，更不用说去破解采用 256 bit 密钥长度的 AES 算法了。

Rijndael 算法由比利时计算机科学家 Vincent Rijmen 和 Joan Daemen 开发，它使用 128 bit、192 bit、256 bit 的密钥长度，比 56 bit 的 DES 更健壮可靠[3]。美国国家标准技术研究所选择 Rijndael 作为美国政府加密标准 AES 的加密算法，取代早期的数据加密标准 DES[1]。Rijndael 作为一种迭代分组加密算法，其数据块长度和密钥长度均是可变的，因此它汇聚了强安全性、高性能、高效率、易用、灵活等优点被广泛应用在各个领域中。

2 Rijndael 算法的设计原理

Rijndael 作为加密标准 AES 算法，其 128 bit 输入分组用以字节为单位的矩阵方阵描述[4]。该数组被复制到 State 数组。数据块长度、密钥长度可以被设定为 128 bit、192 bit、256 bit 三个可选长度，相应的加密轮数分别为 10、12、14，每一轮循环都有一个循环密钥，它来自于初始密钥。

2.1 Rijndael 算法的加密流程

加密过程分为四个阶段：密钥扩展、轮密钥加、$Nr-1$（128 bit、192 bit、256 bit 密钥长度，Nr 分别为 10、12、14）轮变换及最后一轮变换。轮变换包括字节代换、行移位、列混淆和轮密钥加四个过程，最后一轮变换包括字节代换、行移位和轮密钥加三个过程。其流程图如图 1 所示。

(1)状态：指明文分组及每次变换的中间结果[4]，是一个 $4\times Nb$ 的矩阵，Nb 为数据块长度除以 32。

(2)字符代换：用一个简单的查表操作代替了基于矩阵乘法的复杂仿射变换。S 盒是一个 16×16 字节矩阵，包含 8 bit 值所能表达的 256 种可能的变换。把 State 中每个字节的高 4 位作为行值，低 4 位作为列值，取出 S 盒中对应行列的元素作为新的字节输出。

* 基金项目：陕西省自然科学基金(2010JM8035)

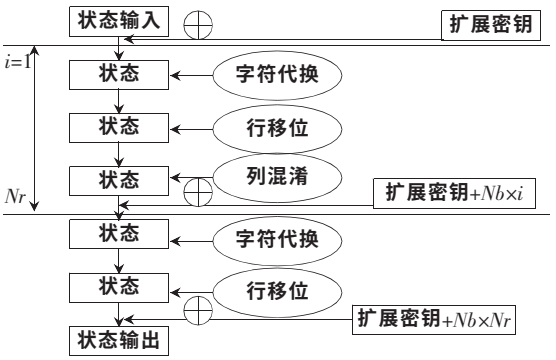

图 1 Rijndael 算法加密流程

(3) 行移位:State 的第一行保持不变,第 2、3、4 行分别循环左移 1、2、3 个字节。

(4)列混淆:可表示为基于系数矩阵 CoefMix 与 State 的矩阵乘法。乘积矩阵中的每个元素 $S[i,j]$是系数矩阵中一行元素 CoefMix $[i,k]$ 与 State 矩阵中对应一列元素 $State[k,j]$的乘积之和。这里的加法与乘法都定义在有限域 GF(28)上:加法即按位"异或"操作,乘法遵循 GF(28)上的多项式乘法规则[2]。

(5)轮密钥加:是基于 State 列的操作,即把 State 一列中的 4 个字节与轮密钥 RoundKey 的 1 个字进行"异或"。

(6)扩展密钥:以 4 个字密钥为输入,生成 44 字扩展密钥数组 $\omega[44]$,为初始轮密钥加阶段和后面 10 轮变换提供轮密钥。输入密钥直接被复制到扩展密钥数组的前 4 个字,然后每次用 4 个字填充扩展密钥数组余下的部分[4]。在扩展密钥数组中,$\omega[i]$值依赖于 $\omega[i-1]$和 $\omega[i-4]$。ω 数组中下标不是 4 的倍数时,$\omega[i]$为 $\omega[i-1]$和 $\omega[i-4]$的"异或"。下标为 4 的倍数时,首先将 $\omega[i-1]$的 4 个字节循环左移 1 个字节,然后利用 S 盒对每个字节进行字节代换,再与轮常量按位"异或"。轮常量是 1 个字,其最右边 3 个字节为 0,最左边 1 个字节的值 $RC[j]$与轮数 j 相关。$RC[1]=1$,$RC[j]=2\cdot RC[j-1]$,乘法定义在 GF(28)上。$RC[j]$值以十六进制表示。

(7)加密轮数 Nr:在 Rijndael 算法中,运算的轮数(Nr)是由 Nb 及 Nk 所决定的[4],轮数的变动定义如表 1 所示。

表 1 Nr、Nb、Nk 关系

Nr	$Nb=4$	$Nb=6$	$Nb=8$
$Nk=4$	10	12	14
$Nk=6$	12	12	14
$Nk=8$	14	14	14

2.2 解密过程

Rijndael 解密过程是加密的逆过程,每轮循环中的步骤都被它们的逆所替换,值得注意的是:循环密钥使用起来应该颠倒次序。

3 Rijndael 算法的应用

Rijindael 算法常被用于文件的加解密过程,加密时先将读入的明文依次分组,用加密密钥将明文加密后写入文件中;解密时用解密密钥将文件中的密文解密后将明文写入结果文件中。

3.1 Rijindael 算法在文件加解密应用的实现

```
try
{                    // 创建新的 Rijndael 对象以产生 Key 和 IV
Rijndael RijndaelAlg= Rijndael.Create();
                     //需要加密的字符串及保存的文件名
string sData = "Here is some data to encrypt.";
string FileName = "CText.txt";
                     //利用 Key 和 IV 加密字符串到文件中
EncryptTextToFile (sData, FileName, RijndaelAlg.Key,
RijndaelAlg.IV);     //利用 Key 和 IV 从文件中解密
string Final=DecryptTextFromFile (FileName, RijndaelAlg.
Key, RijndaelAlg.IV);
Console.WriteLine(Final);                    //显示密码
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
```

3.2 加密模块实现

```
public static void EncryptTextToFile (String Data, String
FileName, byte[] Key, byte[] IV)
{
try
{                                            //创建文件
FileStream fStream =File.Open (FileName, FileMode.
OpenOrCreate);
                                  //创建新的 Rijndael 对象
Rijndael RijndaelAlg=Rijndael.Create();
     //创建加密流,以 passed key 和 initialization vector
                                              (IV)填充
CryptoStream cStream = new CryptoStream(fStream,
RijndaelAlg.CreateEncryptor(Key, IV),
CryptoStreamMode.Write);
                          //用加密流创建 StreamWriter
StreamWriter sWriter = new StreamWriter(cStream);
try
{                                                //加密
sWriter.WriteLine(Data);
}
catch (Exception e)
{
Console.WriteLine("An error occurred:{0}",e.Message);
}
finally
{
sWriter.Close();
```

```
cStream.Close();
fStream.Close();
}
}
catch (CryptographicException e)
{
Console.WriteLine ("A Cryptographic error occurred: {0}", e.Message);
}
catch (UnauthorizedAccessException e)
{
Console.WriteLine ("A file error occurred: {0}", e.Message);
}
}
```

3.3 解密模块的实现

```
public static string DecryptTextFromFile(String FileName, byte[] Key, byte[] IV)
{
try
{                                                    //创建文件流
FileStream fStream = File.Open (FileName, FileMode.OpenOrCreate);
                                                     //创建新的 Rijndael 对象
Rijndael RijndaelAlg = Rijndael.Create();
     //创建加密流,以 passed key 和 initialization vector (IV)填充
CryptoStream cStream = new CryptoStream(fStream, RijndaelAlg.CreateDecryptor(Key, IV), CryptoStreamMode.Read);
                                                     //用加密流创建 StreamWriter
StreamReader sReader = new StreamReader(cStream);
string val = null;
try
{                                                    //解密
val = sReader.ReadLine();
}
catch (Exception e)
{
Console.WriteLine("An error occurred:{0}",e.Message);
}
finally
{
sReader.Close();
cStream.Close();
fStream.Close();
}
return val;                                          //返回密码结果
}
catch (CryptographicException e)
{
Console.WriteLine ("A Cryptographic error occurred: {0}", e.Message);
return null;
}
catch (UnauthorizedAccessException e)
{
Console.WriteLine ("A file error occurred: {0}", e.Message);
return null;
}
}
```

Rijndael 算法一直经受着世界各国密码机构和专家的攻击,最有名的当属 Squre 攻击[4]。目前 Rijndael 算法已被广泛应用于身份认证、数字签名、数据加密等方面,由于硬件的加解密速度要比软件快,且可在物理上保证系统安全,国内很多单位用硬件的方法来实现该算法。

参考文献

[1] 段钢.加密与解密(第 3 版)[M].北京:电子工业出版社,2008.

[2] 冯登国.信息安全中的数学方法与技术[M].北京:清华大学出版社,2009.

[3] Http://zhidao.baidu.com.

[4] DAEMEN J, RIJMAN V. 高级加密标准算法(AES)—Rijndael 的设计[M].谷大武,徐胜波译.北京:清华大学出版社,2003.

(收稿日期:2011-10-10)

作者简介:

张青凤,女,1971 年生,硕士,讲师,主要研究方向:网络安全。

张凤琴,女,1964 年生,在读博士,副教授,主要研究方向:数据库,网络安全。

王蓉,女,1976 年生,在读博士,讲师,主要研究方向:数据库,网络安全。