中图分类号:TP301 文献标识码:A 文章编号:1009-2552(2011)09-0020-03

AES算法及安全性研究

王红珍[1,2]，张根耀[1,2]，李竹林[1,2]

(1. 延安大学计算机学院，延安 716000；2. 延安大学软件研究与开发中心，延安 716000)

摘 要：AES算法是对称密码加密系统中最著名的算法之一，主要介绍AES算法的相关数学基础、算法描述和安全性分析。

关键词：AES；有限域；算法；安全性

AES algorithm and its security

WANG Hong-zhen[1,2], ZHANG Gen-yao[1,2], LI Zhu-lin[1,2]

(1. Computer Science School, Yan'an University, Yan'an 716000, China;
2. Software Research and Development Center, Yan'an University, Yan'an 716000, China)

Abstract: AES algorithm is one of the most famous algorithms for the symmetric encrypt systems. This paper mainly introduces the relevant mathmatics basis, characteristics of the algorithm and the security.

Key words: AES; finite domain; algorthm; security

0 引言

对称密码加密系统中最著名的是数据加密标准DES(Data Encryption Standard)、高级加密标准AES(Advanced Encryption Standard)和国际数据加密标准IDEA(International Data Encryption Algorithm)。1977年,美国国家标准局正式公布并实施了数据加密标准DES,公开了它的加密算法,并批准用于非机密单位和商业上的保密通信。随后,DES成为全世界使用最广泛的加密标准。但是,经过多年的使用,已经发现了DES的很多不足之处,对DES的破解方法也有日趋得逞之势[1]。美国国家标准和技术协会NIST(the National Institute of Standards and Technology)在1997年1月2日正式宣布了公开征集高级加密标准AES。2000年10月2日,NIST宣布Rijndael算法被选中成为将来的AES。Rijndael算法是在1999年下半年,由研究员Joan Daemen和Vincent Rijmen创建的。2001年11月26日,NIST正式公布了新标准AES,其编号为FIPS PUBS197[2]。新标准AES将会代替旧的DES而日益成为加密各种形式的电子数据的实际标准。

1 AES算法的数学基础

有限域就是具有有限个元素的域,元素个数必须是一个素数的幂p^n,n为正整数并称为域的阶,p为素数并称为有限域的特征[3]。

在Rijndael的描述中均使用以2为特征的有限域,并用符号"⊕"表示2特征域中的加法运算。对每一个素数幂恰好存在一个有限域,用$GF(p^n)$表示。当$n=1$时的有限域记为$GF(p)$,其元素可以用集合$\{0,1,\cdots,p-1\}$来表示,该域的两种运算则是"模p的整数加法"和"模p的整数乘法"。

AES算法主要用到两类形式的特征2域上的多项式,即:

$$b(x)=b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0, b_i\in GF(2)$$

$$a(x)=a_3x^3+a_2x^2+a_1x^1+a_0, a_i\in GF(2^8)$$

2 AES算法的描述

Rijndael算法是一个具有可变数据块长度和可变密钥长度的迭代分组密码,其分组长度和密钥长度可分别为128比特、192比特或256比特。但加密时数据一律分组成4行的矩阵(即状态),Rijndael

收稿日期:2011-01-20
基金项目:延安大学专项科研基金(YD2007-21)
作者简介:王红珍(1973-),女,实验师,硕士,研究方向为软件体系结构及应用。

的轮函数就是基于状态的运算。下面对 Rijndael 的三层结构进行分别介绍。

(1)状态、密钥种子和轮数

由于 Rijndael 算法是一个可变数据块长度和可变密钥的迭代加密算法,数据块要经过多次数据变换操作,每一次变换操作产生一个中间结果,称这个中间结果为状态。初始的输入明文可以称为初始状态,状态用一个二维数组表示,其中 S_{rc} 为一个单字节,它有 4 行 N_b 列,其中 r 表示行号,c 表示列号,$0<r<4, 0<c<N_b$。每个元素为一个字节(8 位二进制),如 128 比特的初始状态和 128 比特的密钥各分组为 16 个字节(每字节 8 比特,有 $128=16\times8$)。

密钥种子为 Rijndael 算法的初始密钥,密钥扩展利用它产生算法所需的其它密钥。用 N_b 表示一个数据块中字的个数,用 N_k 表示密钥中字的个数,用 N_r 表示算法轮数,N_r 由 N_b 和 N_k 共同决定。

(2)轮变换

Rijndael 算法的轮变换由四个不同的变换组成,即字节替换(SubBytes)、行移位(ShiftRows)、列混淆(MixColumns)和轮密钥加(AddRoundKey)变换。这些变换可简单的表示为:

```
AddRoundKey(State, RoundKey); //将轮密钥 RoundKey 与明文 State 异或
Round(State, RoundKey) //对前 Nr - 1 轮中的每一轮进行相同的变换
{
 SubBytes(State); // 用 S 盒进行一次变换变换操作
 ShiftRows(State);
 MixColumns(State);
 AddRoundKey(State, RoundKey);
}
最后一轮的密码算法略有不同,即:
FinalRound(State, RoundKey)
{
 SubBytes(State);
 ShiftRows(State);
 AddRoundKey(State, RoundKey);
}//最后的 State 即为密文
```

①字节替换[3]。

字节替换是对状态阵列的字节到字节的变换,它使用字节替换表(S 盒)作用于状态矩阵的每个字节。字节替换有正向字节替换和逆向字节替换,正向字节替换是一个简单的查表操作。AES 定义的 S 盒是由 16×16 个字节组成的矩阵,包含了 8 位值所能表达的 256 种可能的变换。State 中每个字节按照如下的方式映射为一个新的字节:把该字节的高 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输出。

AES 的 S 盒被设计成能防止已有的各种密码分析攻击,Rijndael 开发者特别寻求输入位和输出位之间几乎没有相关性的设计,并且输出值不能通过利用一个简单的数学函数变换输入值所得到。当然,S 盒必须是可逆的,但 S 盒不是自逆的。如,S 盒({95}) = {2A},而逆 S 盒({95}) = {AD}。

②行移位。

行移位是对状态阵列的行到行的变换,也就是将某个字节从一列移到另一列中,它的线性距离是 4 字节的倍数。在这里要注意到,行移位变换确保了某列的四字节被扩展到了 4 个不同的列。正向行移位变换,即 State 的第一行保持不变,把 State 的第二行循环左移一个字节,State 的第三行循环左移两个字节,State 的第四行循环左移三个字节。而逆向行移位变换则是将 State 中的第一行保持不变,后三行执行与正向行移位相反方向的移位操作。

③列混淆。

列混淆是对状态阵列的列到列的变换,它作用于状态阵列的每一列。正向列混淆变换对每列独立地进操作,每列中的每个字节被映射为一个新值,此值由该列中的四个字节通过函数变换得到。列混淆变换可由下面基于 State 的矩阵乘法表示:

$$\begin{bmatrix}02 & 03 & 01 & 01\\ 01 & 02 & 03 & 01\\ 01 & 01 & 02 & 03\\ 03 & 01 & 01 & 02\end{bmatrix}\begin{bmatrix}s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3}\\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3}\\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3}\\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3}\end{bmatrix}=\begin{bmatrix}s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3}\\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3}\\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3}\\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3}\end{bmatrix} \tag{1}$$

逆向列混淆变换可由如下的矩阵乘法定义:

$$\begin{bmatrix}0E & 0B & 0D & 09\\ 09 & 0E & 0B & 0D\\ 0D & 09 & 0E & 0B\\ 0B & 0D & 09 & 0E\end{bmatrix}\begin{bmatrix}s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3}\\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3}\\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3}\\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3}\end{bmatrix}=\begin{bmatrix}s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3}\\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3}\\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3}\\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3}\end{bmatrix} \tag{2}$$

④轮密钥加。

轮密钥加是轮密钥与状态阵列中的对应字节按位异或的变换，轮密钥由密钥扩展得到。在正向轮密钥加变换中，128 位的 State 按位与 128 位的密钥 XOR。把这个操作看成是基于 State 列的操作，即把 State 的一列中的四个字节与轮密钥的一个字进行异或。逆向轮密钥加变换与正向轮密钥加变换相同，因为异或操作是其本身的逆。也可以说由于 AddroundKey 只包括一个异或操作，所以 InvAddRoundKey 与 AddroundKey 变换一致。

(3)密钥的编排

密钥编排方案按照以下准则确定：

①效率：应当能使用尽量少的工作内存来执行该方案；在多种处理器和软件平台上都可方便地实现。

②对称性：应当使用轮常量来消除对称性。

③扩散性：应当将密码密钥的养分有效地扩散到扩展密钥中。

④非线性性：应当具有足够地非线性性，以避免扩展密钥中的差分仅由密码密钥的差分完全决定。

在密钥扩展阶段，将密码密钥扩展为 4 行 $N_b(N_r+1)$ 列的扩展密钥数组，这里用 $W_{4,N_b(N_r+1)}$ 表示。第 i 轮的轮密钥 *ExpandedKey*[i] 由 W 中的第 $N_b * i$ 列到 $N_b * (i+1) - 1$ 列给出：

$$ExpandedKey[i] = W_{\cdot,N_b \cdot i} // W_{\cdot,N_b \cdot i+1} // \wedge // W_{\cdot,N_b \cdot (i+1)-1}, 0 \leqslant i \leqslant N_r \quad (3)$$

密钥扩展函数依赖于 N_k 的值：算法 1 给出了 $N_k < 6$ 的情形，算法 2 给出了 $N_k \geqslant 6$ 的情形。在这两个算法中，W 的前 N_k 列是种子密钥，后面各列由先前的列按递归方式确定。

算法 1　$N_k < 6$ 时的密钥扩展

```
KeyExpansion(byte K[4 * Nk], word W[Nb * (Nr + 1)])
  {
    for (i = 0; i < Nk; i ++)
      w[i] = (key[4i], key[4i + 1], key[4i + 2], key[4i + 3]);
    for(i = Nb; i < Nb(Nr + 1); i ++)
    {
    temp = w[i - 1];
    if( i % Nk == 0)
      temp = SubByte ( RotByte ( temp )) ⊕ Rcon [ i/Nk];
    w[i] = w[i - Nk] ⊕ temp;
        }
  }
```

算法 2　$N_k \geqslant 6$ 时的密钥扩展

```
KeyExpansion(byte K[4 * Nk], word W[Nb * (Nr + 1)])
  {
    for (i = 0; i < Nk; i ++)
      w[i] = (key[4i], key[4i + 1], key[4i + 2], key[4i + 3]);
    for(i = Nb; i < Nb(Nr + 1); i ++)
     {
    temp = w[i - 1];
    if( i % Nk = = 0)
     temp = SubByte ( RotByte ( temp )) Rcon [ i/Nk];
  else if (i % Nk = = 4)
      temp = SubByte( temp);
  w[i] = w[i - Nk] ⊕temp;
    }
  }
```

3　AES 算法的安全性

对于 Rijndael 算法的安全性讨论，通过对以下已知的攻击方法来进行分析，这些攻击有：

①穷举攻击法：Rijndael 算法中最短的密钥长度是 128 比特，如果用穷举法则需要 2^{128} 次，计算量是非常大的，故使用目前技术的穷举法是无效的。

②差分攻击法：这种攻击法是利用大量已知的明文/密文对之间的差异来推测出密钥。对于 Rijndael 算法，当轮数超过三轮，若存在 $1/2^{n-1}$（n 位分组块的）长度可预测性的差异，这种攻击法就可以推测出密钥的位元值，在 Rijndael 算法中已经证明 Rijndael 经过四轮运算后，存在不超过 2^{-150} 比例的可预测性差异，五轮运算后不超过 2^{-300} 比例的可预测差异，因此可以说这种攻击法对 Rijndael 算法是无效的。

③线性攻击法：这种攻击法利用大量收集到的明文/密文对，根据其中可预测的相对关系推导出一个代数展开式，只要能找出相对关系数超过 $2^{n/2}$ 的线性轨迹，就可以找出密钥值。Rijndael 算法已经被证明执行四轮运算后不存在相关系数大于 2^{-75} 的线性轨迹；在执行八轮后，其相关系数大于 2^{-150} 的相关系数也不存在，因此可以说这种攻击法对 Rijndael 算法是无效的。

④平方攻击法：这种攻击是一种明文攻击，攻击强度不依赖于 S 盒、列混合矩阵和密钥扩散准则的选取，但至今为止这种攻击法只能够对经过不超过六次轮运算的 Rijndael 算法有效，（下转第 26 页）

表1 试验参数设置

超时设置	3000ms
重试次数	2
搜索范围	灾备中心所辖所有子网

表2 灾备中心全网扫描

试验次数	路由器数	子网数	子网内设备数	改进后(s)	改进后(s)
1	4	10	148	322	33
2	4	10	130	310	31
3	4	10	126	287	32

表3 设置子网过滤扫描

试验次数	路由器数	子网数	子网内设备数	改进前(s)	改进后(s)
1	4	8	49	69	31
2	4	8	39	58	29
3	4	8	42	59	31

实验分析:在表2实验结果中,未设置子网过滤,未安装SNMP的主机的子网也包含在内,通过SNMP获取主机信息时,采用原算法等待时间较长;在采用本文算法后,使用多线程并行方式,拓扑效率大大提高,可见在含有大量不支持SNMP设备的时候,改进后的算法能极大提高效率。在表3实验结果中,设置了子网过滤,将子网中不支持SNMP的设备过滤后,原算法等待所花的时间虽然大大减少,但是因为改进算法采用的并行策略,在效率上仍然要高出原算法一倍左右。

4 结束语

本文基于数据灾备中心的工作环境,分析了SNMP和ARP的特点,对算法设计思想进行分析,在此基础上对现有的基于ARP和SNMP的网络自动拓扑算法进行改进,利用拓扑过程中的分时特点进行并行化处理,实践应用证明,本算法能够快速有效地在给定搜索范围内完成网络拓扑。

参考文献:

[1] Donnet B. Internet topology discovery: d survey [J]. IEEE Communications Survey and Tutorials, 4th Quarter 2007,9(4):56-69.

[2] Huffaker B, et al. Topology Discovery by Active Probing[C]. Proc. Symp. Applications and the Internet (SAINT), Jan. 2002: 90-96.

[3] Hwa-Chun Lin, Shou-Chuan Lai, Ping-Wen Chen. An algorithm for automatic topology discovery of IP networks[J]. ICC 98-IEEE International Conference on 1998:1192-1196.

[4] Mockapetris P. Domain Names: Concepts and Facilities[S]. IETF, RFC 1034, Nov. 1987.

[5] Mockapetris P. Domain Names: Implementation and Specification [S]. IETF, RFC 1035, Nov. 1987.

[6] David C Plummer. An Ethernet Address Resolution Protocol[S]. IETF, RFC 826, Nov. 1982.

[7] 凌军, 曹阳, 李莉,等. 基于ARP和SNMP的网络拓扑自动发现算法[J]. 武汉大学学报:理学版, 2001,47(1):67-70.

[8] 张旭军, 张长青, 张扬. 剖析第三层交换机[EB/OL]. http://cisco.chinaitlab.com/three/2166.html 2010:10(22).

[9] 邱建林, 何鹏. 一种改进的网络拓扑发现方法[J]. 计算机应用, 2005, 25(4):891-894.

[10] McCloghrie K. Mnagement Information Base for Network Management of TCP/IP Based Internets: MIB-II[S]. IETF, RFC1213, Mar. 1991.

[11] Decker E, Langille P, Rijsinghani A,et al. Definitions of managed objects for bridges[S]. IETF, RFC1493, Jul. 1993.

[12] 彭建, 朱萍, 傅明. 一种基于SNMP协议的网络拓扑发现改进算法[J]. 计算机工程与科学, 2009,31(4):18-20.

责任编辑:么丽萍

(上接第22页)

计算量为$2^{73}+2^{64}$,因此可以说这种攻击法对Rijndael算法是无效的。

⑤内插攻击法:这种攻击法是攻击者利用加密的输入与输出配对,建立一些多项式。如果加密的元件有一个乘法的代数展开式,并且与管理的复杂度结合在一起时,这种攻击便是可行的。攻击的方式是如果攻击者建立的代数展开式的阶度很小,只需要一些加密法的输入和输出配对就可以得到代数展开式的各项系数。但是,Rijndael算法中的F_{28}域,它的展开式为:

$63+8FX^{127}+B5X^{191}+01X^{223}+F4X^{239}+25X^{247}+F9X^{251}+09X^{253}+05X^{254}$,是非常复杂的,因此可以说这种攻击法对Rijndael算法是无效的。

从以上分析可以看出,Rijndael算法对已知的攻击具有较好的免疫性,安全性比较高。

4 结束语

基于对AES算法的数学基础、算法描述、安全性等的研究,在相同的环境以及对同一个大小的文件加(解)密时,AES算法的效率的效率高、保密性好、需要的内存空间少,并且AES算法在所有平台上运行良好,支持并行处理,还可抵抗所有已知攻击。

参考文献:

[1] 曾宪文,高桂革. 对称密码加密系统与公钥密码加密系统[J]. 上海电机学院学报,2005(4):50-52.

[2] 林德敏,林柏钢. 三大密码体制:对称密码、公钥密码和量子密码的理论与技术[J]. 电讯技术,2003(3):6-12.

[3] [美] William Stallings. 密码编码学与网络安全——原理与实践[M]. 3版. 刘玉珍,王丽娜,傅建明,等译. 北京:电子工业出版社,2004.

责任编辑:张禹