仲恺农业技术学院学报，19(2)：44～48，2006
Journal of Zhongkai University of Agriculture and Technology
文章编号：1006－0774(2006)02－0044－05

AES 和 DES 的效率及安全性比较

姚华桢
（仲恺农业技术学院 信息学院，广东 广州 510225）

摘要：运用 C 和 C++ 对数据加密标准（Data Encryption Standard，DES）和高级加密标准（Advanced Encryption Standard，AES）进行了实现，并就两者的效率和安全性进行了比较，证明了 AES 算法的优越性。
关键词：AES；DES；效率；安全
中图分类号：TP309.7　　文献标识码：A

The comparison of efficiency and security between AES and DES

YAO Hua-zhen
（College of Information，Zhongkai University of Agriculture and Technology，Guangzhou 510225，China）

Abstract：The implement of two algorithm，Data Encryption Standard（DES）and Advanced Encryption Standard（AES）were discussed. Compared with DES，AES has more superiority in efficiency and security.
Key words：AES；DES；efficiency；security

1973 年 5 月 15 日，美国国家标准局在联邦记录中公开征集密码体制，这一举措导致了数据加密标准（Data Encryption Standard，DES）算法的出现，并且一度成为世界上最广泛使用的密码体制. 但后来由于 DES 算法及其变形的安全强度已经难以继续满足新的安全需要，难以对抗 20 世纪末出现的差分和线性密码分析（目前最强有力的密码破解工具），而且其实现速度，代码大小以及跨平台性均难以满足新的需求，于是美国政府于 1997 年开始公开征集新的数据加密标准（Advanced Encryption Standard，AES）算法，以取代 DES. 经过三轮筛选，最后选中比利时密码学家 Joan Daemen 和 Vincent Rijmen 提出的密码算法 Rijndael 作为 AES 正式取代 DES[1].

AES 算法不但高效、简洁，且最重要的是 AES 算法能够对抗所有已知类型的攻击. 目前，对数据的加密逐步由 AES 算法取代 DES 算法，特别是对实时性要求高的数据，比如视频数据. 作者运用 C 和 C++ 不仅对 DES 进行了软件实现，而且对 AES 的普通实现算法以及快速实现算法分别做了实现，并比较 DES 算法和 AES 算法的加解密效率以及 AES 算法的两种实现方法的加解密效率，分析 DES 算法和 AES 算法的安全性差异，以期证明 AES 算法的优越性.

1　DES 算法描述

DES 对 64 位的明文分组进行操作，密钥长度为 8 个字节. 通过一个初始置换，将明文分成左半部分和右半部分，各 32 位长. 然后进行 16 轮完全相同的运算，这些运算被称为函数 f，在运算过程中数据与密钥结合. 经过 16 轮后，左、右半部分合在一起经过一个末置换（初始置换的逆置换），即算完成该算法[4].

在每一轮中，密钥位移位，然后再从密钥的 56 位中选出 48 位. 通过一个扩展置换将数据的右半部分扩展成 48 位，并通过一个异或操作与 48 位密钥结合，通过 8 个 S 盒将这 48 位替代成 32 位数据，再将其

收稿日期：2005－12－20
作者简介：姚华桢（1980－），女，江西南昌人，助教，硕士.
?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

置换一次. 以上四步运算构成了函数 f. 然后，通过另一个异或运算，函数 f 的输出与左半部分异或，其结果即成为新的右半部分，原来的右半部分成为新的左半部分. 将该操作重复 16 次，便实现了 DES 的 16 轮运算.

假设 B_i 是第 i 次迭代的结果，L_i 和 R_i 是 B_i 的左半部分和右半部分，K_i 是第 i 轮 48 位密钥，且 f 是实现代替、置换及密钥异或等运算的函数，那么每一轮就是：$\begin{matrix} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f\ (R_{i-1},\ K_i) \end{matrix}$. 具体步骤如下：

（1）在第一轮运算之前执行，对输入分组实施的初始置换.

（2）密钥置换：为确保密钥不发生错误，DES 的密钥中每个字节的第 8 位用来作为奇偶校验，故 DES 实际密钥长度为 56 位. 在 DES 的每一轮中，从 56 位密钥产生出不同的 48 位子密钥.

（3）扩展置换：扩展运算将数据的右半部分 R，从 32 位扩展到了 48 位. 由于这个运算改变了位的次序，重复了某些位，故被称为扩展置换.

（4）S 盒替代：将扩展分组与压缩后的密钥异或，然后将 48 位的结果输入，进行替代运算. 替代由 8 个 S 盒完成. 每个 S 盒都有 6 位输入，4 位输出，且这 8 个 S 盒是不同的. 48 位的输入被分为 8 个 6 位的分组，每一分组对应一个 S 盒代替操作.

（5）P 盒置换：S 盒替代运算后的 32 位输出按照 P 盒进行置换. 将 P 盒置换的结果与最初的 64 位分组的左半部分异或，然后左、右半部分交换，接着开始另一轮.

（6）末置换：末置换是初始置换的逆置换，在 DES 最后一轮置换后，左半部分和右半部分并未交换，而是将 R_{16} 与 L_{16} 并在一起形成一个分组作为末置换的输入.

2　AES 算法描述

Rijndael 是具有可变分组长度和可变密钥长度的分组密码[1]. 其分组长度和密钥长度均可独立地设定为 32 bit 的任意倍数，最小值为 128 bit，最大值为 256 bit. 其输入输出均可看为 8 位字节的一维数组：假设输入明文为 p_0，$p_1 \cdots p_i \cdots$　$0 \leqslant i \leqslant 4 * N_b - 1$. 将明文映射到一个字节矩阵上，称之为状态（图 1）.

p_0	p_4	p_8	p_{12}
p_1	p_5	p_9	p_{13}
p_2	p_6	p_{10}	p_{14}
p_3	p_7	p_{11}	p_{15}

图 1　数据块长度为 128 bits 的状态

Fig. 1　Block data state of 128 bits in length

本例中 $N_b = 4$，密钥同理可映射到密钥状态中.

Rijndael 加密算法由 3 部分组成：①一个初始轮密钥加法变换；②$N_r - 1$ 轮变换；③最后一轮变换. 用伪 C 码表示为：

```
Rijndael (State, CipherKey)
{ KeyExpansion (CipherKey, ExpandKey) //密钥扩展，即把输入的密钥扩展为加密用的密钥
  AddRoundKey (State, ExpandKey)          //初始轮密钥加法变换
  for (i=1; i<Nr; i++)                    //Nr-1 轮轮变换
     Round (State, ExpandedKey [i] )
     {      ByteSub (State);              //S-盒变换，非线性的砖匠置换
            ShiftRow (State);             //字节换位，将状态中的行按不同的偏移量进行循环移位
            MixColumn (State);            //作用在状态各列的砖匠变换
            AddRoundKey (State, ExpandedKey [i];)   //密钥加法      }
  FinalRound (State, ExpandedKey [Nr] )
     {      ByteSub (State);              //同上
```

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

```
        ShiftRow (State);                    //同上
    AddRoundKey (State, ExpandedKey [Nr]); //同上      }          }
```

要在 32 位或者更长的处理器上快速实现 AES 算法，可以考虑将轮变换的不同步骤合并为一组表的查询[1]. 下面将详细说明如何做到这一点.

假设轮变换的输入用 a 表示，而 SubBytes 的输出用 b 表示，则有：

$$b_{i,j} = S_{RD}[a_{i,j}], 0 \leqslant i < 4; 0 \leqslant j < N_b \tag{1}$$

又设 ShiftRows 的输出用 c 表示，而 MixColumns 的输出用 d 表示：

$$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j+c0} \\ b_{1,j+c1} \\ b_{2,j+c2} \\ b_{3,j+c3} \end{bmatrix}, 0 \leqslant j < N_b \tag{2}$$

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}, 0 \leqslant j < N_b \tag{3}$$

(2)式中下标的加法必须是 mod N_b 的. (1)~(3)式可以合并为：

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} S_{RD}[a_{0,j+c_0}] \\ S_{RD}[a_{1,j+c_1}] \\ S_{RD}[a_{2,j+c_2}] \\ S_{RD}[a_{3,j+c_3}] \end{bmatrix}, 0 \leqslant j < N_b \tag{4}$$

上述的矩阵乘法运算可以看作是 4 个列向量之间的线性组合：

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} S_{RD}[a_{0,j+c_0}] \oplus \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} S_{RD}[a_{1,j+c_1}] \oplus \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} S_{RD}[a_{2,j+c_2}] \oplus \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} S_{RD}[a_{3,j+c_3}], 0 \leqslant j < N_b \tag{5}$$

定义 4 个 T 表：T_0、T_1、T_2 以及 T_3：

$$T_0[a] = \begin{bmatrix} 02 \cdot S_{RD}[a] \\ 01 \cdot S_{RD}[a] \\ 01 \cdot S_{RD}[a] \\ 03 \cdot S_{RD}[a] \end{bmatrix}, \quad T_1[a] = \begin{bmatrix} 03 \cdot S_{RD}[a] \\ 02 \cdot S_{RD}[a] \\ 01 \cdot S_{RD}[a] \\ 01 \cdot S_{RD}[a] \end{bmatrix} \tag{6}$$

$$T_2[a] = \begin{bmatrix} 01 \cdot S_{RD}[a] \\ 03 \cdot S_{RD}[a] \\ 02 \cdot S_{RD}[a] \\ 01 \cdot S_{RD}[a] \end{bmatrix}, \quad T_3[a] = \begin{bmatrix} 01 \cdot S_{RD}[a] \\ 01 \cdot S_{RD}[a] \\ 03 \cdot S_{RD}[a] \\ 02 \cdot S_{RD}[a] \end{bmatrix} \tag{7}$$

每一个 T 表都有 256 个 4 字节的条目，从而需要 4 KB 的存储空间. 为了使用这些表，把(5)式改写成下面的表达：

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = T_0[a_{0,j+c_0}] \oplus T_1[a_{1,j+c_1}] \oplus T_2[a_{2,j+c_2}] \oplus T_3[a_{3,j+c_3}] \oplus, 0 \leqslant j < N_b \quad (8)$$

考虑到 AddRoundKey 可以通过在每一列上执行一个额外的 32 位异或运算来实现，可以用 4 KB 的表来实现一个查表. 该实现方案对于每一轮的每一列仅需要 4 次表查询和 4 次异或运算. 实现这些操作的效率很高. 同理，在解密算法中，也可以将轮变换的不同步骤合并为一组表的查询.

3　AES 的两种实现方法的速度比较

作者用 AES 算法的普通实现和快速实现算法对 3.77 M（3 960 928 字节）的文件进行加密. 程序的运行环境为Windows 2000，CPU 2.2G HZ，内存 256 M. AES 分组长度为 128 比特，密钥长度为 128 比特. 加密和解密时间测试结果如表 1 所示：

以上测试可以看出 AES 快速实现的加解密速度将近为普通实现时的两倍. 下面比较一下两者的加密效率（表 2）：

表 1　两种实现时的加解密的时间比较

Tab. 1　Time comparison of fast implementation and general implementation

实现方法 Implementation method	加密 Encrypt/s	解密 Decrypt/s
普通实现 General implementation	6	10
快速实现 Fast implementation	3	5

表 2　两种实现时的加解密效率比较

Tab. 2　Efficiency comparison of fast implementation and general implementation

实现方法 Implementation method	加密 Encrypt/Mb/s	解密 Decrypt/Mb/s
普通实现 General implementation	3.77× 8/6；5.027	3.77× 8/10= 3.016
快速实现 Fast implementation	3.77× 8/3；10.053	3.77× 8/5= 6.032

4　AES 算法与 DES 算法的比较

4.1　加密效率的测试比较

作者实现了 DES 算法，然后也对 3.77M（3 960 928 字节）的文件进行加密. 程序的运行环境同样为 Windows 2000，CPU 2.2G HZ，内存 256M. 其中 DES 分组长度为 64 比特，密钥长度为 56 比特. 对 DES 算法做了长期的测试. 其加密和解密时间测试结果见表 3：

以上测试可以看出 AES 在加密速度上明显优于 DES，而且 AES 的密钥长度是 DES 密钥长度的两倍. 下面比较一下两者的加密效率（表 4）：

表 3　AES 和 DES 加解密的时间比较

Tab. 3　Encryption time comparison of AES and DES

算法 Algorithm	加密 Encrypt/s	解密 Decrypt/s
DES	18	18
AES	6	10

表 4　AES 和 DES 的加解密效率比较

Tab. 4　Encryption efficiency comparison of AES and DES

算法 Algorithm	加密 Encrypt/Mb/s	解密 Decrypt/Mb/s
DES	3.77× 8/18≈1.676	3.77× 8/18≈1.676
AES	3.77× 8/6≈5.027	3.77× 8/10= 3.016

4.2　安全性比较

自 DES 算法 1977 年首次公诸于世以来，学术界对其进行了深入的研究，围绕它的安全性展开了激烈的争论[1]. 在技术上，对 DES 的批评主要集中在以下 3 个方面：

（1）作为分组密码，DES 的加密单位仅有 64 位二进制，这对于数据传输来说太小，因为每个分组仅含 8 个字符，而且其中某些位还要用于奇偶校验或其他通讯开销.

（2）DES 的密钥的位数太短，只有 56 比特，而且各次迭代中使用的密钥 k_i 是递推产生的，这种相关

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

必然降低密码体制的安全性. 在现有技术下用穷举法寻找密钥已趋于可行. 1999 年在电子前沿组织（Electronic Frontier Foundation, EFF）进行的一次测试中, 只用了不到 3 d 的时间就破解了一个 DES 加密系统. 所以若要保护 10 年以上的数据最好不要用 DES 算法

（3）DES 不能对抗差分和线性密码分析.

相对 DES 算法来说, AES 算法无疑解决了上述问题, 主要表现在如下几方面:

（1）AES 标准支持可变分组长度, 分组长度可设定为 32 比特的任意倍数, 最小值为 128 比特, 最大值为 256 比特.

（2）AES 的密钥长度比 DES 大, 它也可设定为 32 比特的任意倍数, 最小值为 128 比特, 最大值为 256 比特. 用穷举法是不可能破解的. 假设取密钥长度为 128 比特, 那么要破解 2^{128} 个可能的密钥的密文平均要尝试的密钥的个数为 $\frac{1}{2^{128}}\sum_{i=1}^{2^{128}} i \approx 2^{127}$ 个. 也就是要进行 2^{127} 次解密运算, 根据表 4 中的测试数据可以算出一个 AES 分组解密需要的时间为: 128/（5.027×10^6）≈2.55×10^{-5} 秒. 由此可以算出一台 PC 机用穷举法破解大概需要的时间为:

$$2^{127} \times 2.55 \times 10^{-5} \approx 2.752 \times 10^{26} \text{年}$$

可以看出这实际上是不可行的. 在可预计的将来, 如果计算机的运行速度没有根本性的提高, 用穷举法破解 AES 密钥几乎不可能.

（3）AES 算法的设计策略是宽轨迹策略（Wide Trail Strategy, WTS）[1]. WTS 是针对差分分析和线性分析提出的, 可对抗差分密码分析和线性密码分析.

5 结论

用软件实现 DES 算法和 AES 算法, 后者的效率明显高于前者, 而且由于 AES 算法的简洁性, 使得它的实现更为容易. AES 作为新一代的数据加密标准, 其安全性也远远高于 DES 算法. 更为重要的是, AES 算法硬件实现的速度大约是软件实现的 3 倍, 这就给用硬件实现加密提供了很好的思路.

参考文献:

[1] JOAN Daemen, VINCENT Rijmen. 高级加密标准(AES)算法——Rijndael 的设计[M]. 谷大武, 徐胜波, 译. 北京: 清华大学出版社, 2003: 31－64.

[2] 陈爱民. 计算机的安全与保密[M]. 北京: 电子工业出版社, 1992: 22－54.

[3] ANDREW S, TANENBAUM. 计算机网络[M]. 潘爱民, 泽. 北京: 清华大学出版社, 2002: 447－480.

[4] 卢开澄. 计算机密码学——计算机网络中的数据保密与安全[M]. 北京: 清华大学出版社, 1998: 6－62.

【责任编辑 曾东发】

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net