第32卷 第21期 计 算 机 工 程 2006年11月
Vol.32 No.21 Computer Engineering November 2006
·安全技术· 文章编号：1000—3428(2006)21—0173—02 文献标识码：A 中图分类号：TP309

高级加密标准Rijndael算法中S盒的替换方案

殷新春[1,2]，杨 洁[1]
(1. 扬州大学计算机科学与工程系，扬州 225009；2. 南京大学计算机软件新技术国家重点实验室，南京 210093)

摘 要：分析了高级加密标准Rijndael算法中非线性变换S盒的设计思想，对S盒构造过程中的仿射变换加以改变构造出了一批密码性能良好的8×8的S盒，从方差的角度分析了S盒的雪崩概率，并从中得到部分规律，这将有助于寻找更加安全的S盒。
关键词：高级加密标准；Rijndael；S盒；仿射变换

Scheme of Replacement of S-box in AES Rijndael Algorithm

YIN Xinchun[1,2], YANG Jie[1]
(1.Department of Computer Science and Engineering, Yangzhou University, Yangzhou 225009;
2. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

【Abstract】 The constitution of nonlinear transformation S-box in AES Rijndael algorithm is analyzed. The affine permutation in the process of S-box construction is changed in order to construct a kind of 6×6 S-box with some good cryptographic properties, and the avalanche probabilities of these S-boxes are analyzed in the light of variance, and some rules are obtained from the analysis, which will help to find safer S-boxes.
【Key words】 AES; Rijndael; S-box; Affine permutation

S盒是许多密码算法中的唯一非线性部件，因此，它的密码强度决定了整个密码算法的安全强度[1]。S盒的构造方法有很多，如随机选取并测试，使用数学函数等[2]。美国国家标准技术局(NIST)在2001年发布了高级加密标准(AES)。高级加密标准Rijndael算法中的S盒是通过$GF(2^8)$上的乘法逆与一个可逆仿射变换合成的，它满足若干密码学性质，能够抵抗现有的各种攻击[3]。

本文根据AES S盒的设计思想，构造了一批满足AES S盒的设计准则的8×8的S盒，这类S盒可以替换高级加密标准Rijndael算法中的S盒而不会减弱Rijndael算法的安全性。

此外，如果从多个此类8×8的S盒出发，可以构造出一批规模更大或较小的S盒。这些S盒对进一步设计密码算法提供了非线性资源。

1 AES S盒的设计思想及其性能

1.1 设计思想

字节代换SubBytes是Rijndael密码中惟一的非线性变换。它包含一个作用在状态字节上的S盒。AES S盒的构造可用矩阵的形式表示如下[4-6]：

$$\begin{bmatrix} z_7 \\ z_6 \\ z_5 \\ z_4 \\ z_3 \\ z_2 \\ z_1 \\ z_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad (1)$$

式(1)中的向量Y表示S盒输入向量X的逆映射。若将矩阵U简写，成如下形式：

$U=[U_7\ U_6\ U_5\ U_4\ U_3\ U_2\ U_1\ U_0]$

其中，每个U_i都是一个8维的列向量。

上式是将S盒的一个输入字节分2步进行变换：首先取其在$GF(2^8)$上的乘法逆，然后将其逆元通过一个仿射变换得到了S盒的一个输出字节。而仿射变换又可以写成如下的多项式形式：

$$b(x) = v(x) + (X^{-1})\ u(x) \bmod m(x) \quad (2)$$

其中，$m(x)=x^8+1$，其形式是最简单的，而$u(x)$与$m(x)$是互素的，$v(x)$的选择使得S盒没有不动点$(S(x)=x)$和反不动点$(S(x)=\overline{x})$[4]。从式(1)可以很容易看出$v(x)=(x^6+x^5+x+1)$，下面就对$u(x)$进行求解：

$$[U_7\ U_6\ U_5\ U_4\ U_3\ U_2\ U_1\ U_0]\cdot[0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T = U_0 = u(x)\cdot 1 = x^4+x^3+x^2+x+1 \quad (3)$$

经过实验验证，将式(2)和有限域上的求逆运算结合起来可以生成AES S盒中的256个元素。

定理1 当$m(x)=x^n+1$时，由多项式$u(x)$转换成的矩阵U有以下形式：

$U=[U_7\ U_6\ U_5\ U_4\ U_3\ U_2\ U_1\ U_0]$

其中，$U_7=(U_6$循环左移1bit)，$U_6=(U_5$循环左移1bit)，…，$U_1=(U_0$循环左移1bit)。

证明：U_i的多项式表示为$u(x)\cdot x^i$；

U_{i-1}的多项式表示为$u(x)\cdot x^{i-1}$，

因为$u(x)\cdot x^i=(x\cdot(u(x)\cdot x^{i-1}))\bmod(x^n+1)$，所以

(1)当U_{i-1}的最高位为0时，$U_i=(U_{i-1}<<1)\Leftrightarrow U_{i-1}$循环左移1bit；

(2)当U_{i-1}的最高位为1时，$U_i=(U_{i-1}<<1)\oplus(100\ldots01)\Leftrightarrow U_{i-1}$循环左移1bit。

根据定理1可将矩阵U和向量U·Y表示成如下形式：

基金项目：国家自然科学基金资助项目(60473012)
作者简介：殷新春(1962－)，男，博士、教授，主研方向：并行与分布计算，信息安全；杨 洁，硕士生
收稿日期：2006-02-24 **E-mail**：yzgg_100@163.com

$$[U_7\ U_6\ U_5\ U_4\ U_3\ U_2\ U_1\ U_0] = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 \\ u_7 & u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \\ u_6 & u_7 & u_0 & u_1 & u_2 & u_3 & u_4 & u_5 \\ u_5 & u_6 & u_7 & u_0 & u_1 & u_2 & u_3 & u_4 \\ u_4 & u_5 & u_6 & u_7 & u_0 & u_1 & u_2 & u_3 \\ u_3 & u_4 & u_5 & u_6 & u_7 & u_0 & u_1 & u_2 \\ u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_0 & u_1 \\ u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_0 \end{bmatrix} \xrightarrow{U \cdot Y} [U_7\ U_6\ U_5\ U_4\ U_3\ U_2\ U_1\ U_0] \cdot Y = \begin{bmatrix} u_0y_7+u_1y_6+u_2y_5+u_3y_4+u_4y_3+u_5y_2+u_6y_1+u_7y_0 \\ u_7y_7+u_0y_6+u_1y_5+u_2y_4+u_3y_3+u_4y_2+u_5y_1+u_6y_0 \\ u_6y_7+u_7y_6+u_0y_5+u_1y_4+u_2y_3+u_3y_2+u_4y_1+u_5y_0 \\ u_5y_7+u_6y_6+u_7y_5+u_0y_4+u_1y_3+u_2y_2+u_3y_1+u_4y_0 \\ u_4y_7+u_5y_6+u_6y_5+u_7y_4+u_0y_3+u_1y_2+u_2y_1+u_3y_0 \\ u_3y_7+u_4y_6+u_5y_5+u_6y_4+u_7y_3+u_0y_2+u_1y_1+u_2y_0 \\ u_2y_7+u_3y_6+u_4y_5+u_5y_4+u_6y_3+u_7y_2+u_0y_1+u_1y_0 \\ u_1y_7+u_2y_6+u_3y_5+u_4y_4+u_5y_3+u_6y_2+u_7y_1+u_0y_0 \end{bmatrix}$$

从右端的矩阵可以看出：$[U_7\ U_6\ U_5\ U_4\ U_3\ U_2\ U_1\ U_0] \cdot Y = u_0 \cdot Y \oplus u_1 \cdot (Y$ 循环左移 1bit) $\oplus u_2 \cdot (Y$ 循环左移 2bit) $\oplus u_3 \cdot (Y$ 循环左移 3bit) $\oplus \ldots \oplus u_7 \cdot (Y$ 循环左移 7bit)，$u_i = 0$ 或 1。

矩阵的这一特点使得构造 S 盒的程序运行起来速度很快，因为只要将乘法逆元集中的每个元素进行循环移位和异或运算即可。

1.2 性能分析

通过对 AES 的 S 盒进行测试，得到如下的结果：

(1)满足整数平衡性；

(2)非线性度 $N_s = 112$；

(3)差分均匀度 $\delta = 4/256$；

(4)雪崩概率：虽然 S 盒的每个输出比特的布尔函数都不满足 SAC，但 S 盒的输出比特的雪崩概率都很接近 1/2，具体数据见表 1。

表 1 AES S 盒的雪崩概率

取补比特位	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
00000001	0.515 625	0.515 625	0.453 125	0.562 5	0.453 125	0.484 375	0.453 125	0.5
00000010	0.468 75	0.484 375	0.562 5	0.5	0.484 375	0.453 125	0.5	0.531 25
00000100	0.515 625	0.515 625	0.5	0.468 75	0.562 5	0.5	0.531 25	0.5
00001000	0.531 25	0.531 25	0.468 75	0.453 125	0.5	0.531 25	0.5	0.546 875
00010000	0.453 125	0.5	0.453 125	0.515 625	0.5	0.5	0.546 875	0.531 25
00100000	0.453 125	0.515 625	0.515 625	0.468 75	0.468 75	0.546 875	0.531 25	0.531 25
01000000	0.531 25	0.531 25	0.468 75	0.515 625	0.468 75	0.531 25	0.531 25	0.484 375
10000000	0.515 625	0.562 5	0.515 625	0.531 25	0.484 375	0.531 25	0.484 375	0.515 625

2 AES S 盒的替换方案

寻找密码性能良好的 S 盒可以从构造 S 盒的求逆变换和仿射变换出发。

在有限域上求逆的变换保证了 S 盒的非线性性。$GF(2^8)$ 上的 8 次既约多项式有 30 个[7]，相应的乘法逆元集就有 30 个。本文所举出的例子仍然使用 AES 中的既约多项式。

可逆仿射变换中的 $m(x)$的选择只要满足最高次数是 8 次即可。AES 算法选取的是形式最简单的 $m(x)$，但它是一个可约的多项式。要找到与 $m(x)$互素的多项式 $u(x)$，首先要找出 $m(x)$的所有多项式因子，显然，$m(x) = x^8 + 1 = (x+1)^8$，所以 $u(x)$只要满足这样一个条件即可：不含$(x+1)$这个因式。由于含$(x+1)$这个因式的多项式都有偶数个非 0 系数[8]，因此 $u(x)$只要满足项数个数是奇数这个条件即可。假定选择的 $m(x)$是不可约多项式，则 $u(x)$的选取可以是任意的，因为不可约多项式与任何多项式都互素。

下面所举出的例子仍然使用 AES 中的 $m(x)$，则 $u(x)$以及与之对应的 $v(x)$分别为：

$u(x) = x^6 + x^4 + x^2 + x + 1$，　$v(x) = x^7 + x^5 + x^2 + x$；

$u(x) = x^6 + x^5 + x^3 + x^2 + 1$，　$v(x) = x^6 + x^5 + x^4 + x^2 + x$；

$u(x) = x^6 + x^5 + x^4 + x + 1$，　$v(x) = x^6 + x^5 + x^2$；

$u(x) = x^7 + x^3 + x^2 + x + 1$，　$v(x) = x^6 + x^5 + x^2 + x$；

由于 $v(x)$的选择只要使得 S 盒没有不动点和反不动点，因此对于每个满足条件的 $u(x)$，都可以有若干个满足条件的 $v(x)$与之对应，从而一个 $u(x)$就能生成若干个密码性能与 AES S 盒相当的 8×8 的 S 盒。

3 S 盒雪崩概率分析

根据 AES S 盒的设计思想，构造了一批密码性能良好的 8×8 的 S 盒，用这种方法所构造出来的 S 盒除了雪崩概率有所改变外，其他密码学性能都与 AES S 盒一样。

S 盒的雪崩概率是指改变输入的 1 比特，输出比特改变的概率[1]。当概率为 0.5 时是最理想的。本文从方差的角度去分析 S 盒的雪崩概率。根据方差的定义，方差越小，S 盒的雪崩概率集中在 0.5 附近的程度就越高，安全性也就越好。

最高次数为 7 且项数个数为 5 的多项式一共有 35 个，通过对这 35 个 $u(x)$进行穷举测试，发现 S 盒的雪崩概率不受 $v(x)$的影响。以下的结论是针对 $m(x) = x^8 + 1$ 而言的：

(1)使构造出来的 S 盒雪崩概率方差的绝对值为 0.000 73 的 $u(x)$：

$u(x) = x^7 + x^5 + x^3 + x^2 + 1$；　$u(x) = x^7 + x^5 + x^4 + x^2 + 1$；

$u(x) = x^7 + x^5 + x^4 + x^2 + x$；　$u(x) = x^7 + x^6 + x^4 + x^2 + x$；

$u(x) = x^7 + x^6 + x^4 + x^3 + x$；

(2)使构造出来的 S 盒雪崩概率方差的绝对值为 0.001 45 的 $u(x)$：

$u(x) = x^7 + x^5 + x^2 + x + 1$；　$u(x) = x^7 + x^6 + x^4 + x + 1$；

$u(x) = x^7 + x^6 + x^5 + x^3 + 1$；　$u(x) = x^7 + x^6 + x^5 + x^4 + x$；

$u(x) = x^7 + x^5 + x^4 + x^3 + x^2$；

(3)使构造出来的 S 盒雪崩概率方差的绝对值为 0.001 71 的 $u(x)$：

$u(x) = x^7 + x^5 + x^3 + x + 1$；　$u(x) = x^7 + x^6 + x^4 + x^2 + 1$；

$u(x) = x^7 + x^5 + x^3 + x^2 + x$；　$u(x) = x^7 + x^5 + x^4 + x^3 + x$；

$u(x) = x^7 + x^6 + x^5 + x^3 + x$；

(4)使构造出来的 S 盒雪崩概率方差的绝对值为 0.003 58 的 $u(x)$：

$u(x) = x^7 + x^4 + x^2 + x + 1$；　$u(x) = x^7 + x^6 + x^3 + x + 1$；

$u(x) = x^7 + x^6 + x^5 + x^2 + 1$；　$u(x) = x^7 + x^4 + x^3 + x^2 + x$；

$u(x) = x^7 + x^6 + x^5 + x^4 + x^2$；

(5)使构造出来的 S 盒雪崩概率方差的绝对值为 0.005 07 的 $u(x)$：

$u(x) = x^7 + x^4 + x^3 + x + 1$；　$u(x) = x^7 + x^5 + x^4 + x + 1$；

$u(x) = x^7 + x^5 + x^4 + x^3 + 1$；　$u(x) = x^7 + x^6 + x^3 + x^2 + x$；

$u(x) = x^7 + x^6 + x^5 + x^3 + x^2$；

(6)使构造出来的 S 盒雪崩概率方差的绝对值为 0.005 89 的 $u(x)$：

$u(x) = x^7 + x^3 + x^2 + x + 1$；　$u(x) = x^7 + x^6 + x^2 + x + 1$；

$u(x) = x^7 + x^6 + x^5 + x + 1$；　$u(x) = x^7 + x^6 + x^5 + x^4 + 1$；

$u(x) = x^7 + x^6 + x^5 + x^4 + x^3$；

(7)使构造出来的 S 盒雪崩概率方差的绝对值为 0.011 07 的 $u(x)$：

(下转第 176 页)

(3)将找到的这条边做记号，并从找到的这个点(设为 u_i)出发，重复(2)。如果向下继续搜索需要条件，譬如，需要 u_2 同时满足授权给某个点时，就要搜索 u_2 看其是否授权即可，然后判断 u_i 是否能继续向下进行下去。直到 u_i 没有指向它的边为止，将 u_i 从 u 中删除，然后返回上一个点，看是否有其它路径，如有，则继续搜索，如无，则将那个点继续删除，返回直至 u_0 为止。

(4)重新从 u_0 指向它的边中选择一条未做标记的边，重复(2)、(3)，一旦发现有记号的路，则放弃选择另一条继续。

(5)直到指向 u_0 的路都有标记为止。

具体实现，如图 1 所示。

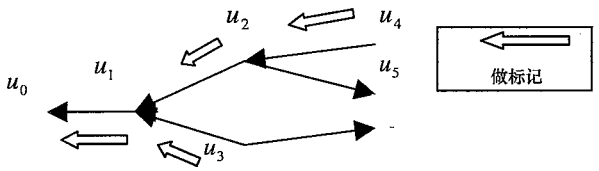

图 1 搜索不成功

搜索过程(图 2)如下：

$\{u_0\} \rightarrow \{u_0,u_1\} \rightarrow \{u_0,u_1,u_2\} \rightarrow \{u_0,u_1,u_2,u_4\} \rightarrow \{u_0,u_1,u_2\} \rightarrow \{u_0,u_1\} \rightarrow \{u_0,u_1,u_3\} \rightarrow \{u_0,u_1\} \rightarrow \{u_0\}$

最后输出不成功。

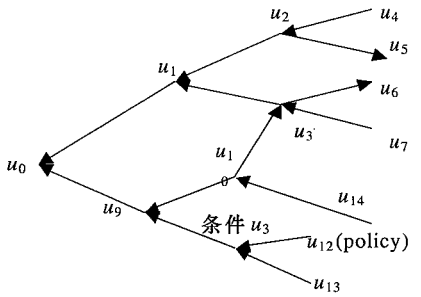

图 2 搜索成功

$\{u_0\} \rightarrow \{u_0,u_1\} \rightarrow \{u_0,u_1,u_2\} \rightarrow \{u_0,u_1,u_2,u_4\} \rightarrow \{u_0,u_1,u_2\} \rightarrow \{u_0,u_1\} \rightarrow \{u_0,u_1,u_3\} \rightarrow \{u_0,u_1,u_3,u_7\} \rightarrow \{u_0,u_1,u_3\} \rightarrow \{u_0,u_1,u_3,u_{10}\} \rightarrow \{u_0,u_1,u_3,u_{10},u_{14}\} \rightarrow \{u_0,u_1,u_3,u_{10}\} \rightarrow \{u_0,u_1,u_3\} \rightarrow \{u_0,u_1\} \rightarrow \{u_0\} \rightarrow \{u_0,u_9\} \rightarrow \{u_0,u_9,u_{10}\} \rightarrow \{u_0,u_9\} \rightarrow \{u_0,u_9,u_{11}\} \rightarrow \{u_0,u_9,u_{11},u_{12}\}$

最后输出成功。

3 算法的比较

令 n 为顶点数，e 为边数，此算法的时间复杂度为 $O(n^3)$，因为在最坏情况，就是每次要搜索所有的边，其时间复杂度为 $O(n^2)$，在假设每个点都带条件的话，就是 $O(n^2 \times n)$，即时间复杂度为 $O(n^3)$。而原算法最终的时间复杂度为 $O(mn^2(mns)^c)$，所以比原算法要快，并且空间复杂度为 O(n+2e) 也很小，比原算法提高了效率。

对于整个过程，每读入一条断言都会使图中的点以及授权关系动态地改变。所以每当输入否定凭证的断言时，原算法仅能处理满足单调性的策略断言，不能删除其他断言已写入黑板的接收记录，而此新算法只需将对应的授权关系的边删除，然后重新搜索，解决了否定安全凭证的问题。

存储图中的点与边可以用 2 种方式来存储：邻接矩阵和邻接表。当图为稠密图时用邻接矩阵；而当图为稀疏图时，用邻接矩阵存储空间浪费很大，所以用邻接表更好。对上述算法，如与请求 r 一致的凭证很多时，就用邻接矩阵，将其存储空间设为足够大。如与请求 r 一致的凭证很少时，用邻接多重表，将点、邻接表和逆邻接表一起记录下来。

4 结束语

本文介绍了有关信任管理模型的基本概念，提出了的新的一致性验证算法，解决了不可撤消性，并使其更加简便快捷。但该算法还有一些不足，比如没有将所有可能出现的情况考虑到算法中去以及难以处理不确定的安全信息等问题。

参考文献

1 Blaze M, Feigenbaum J, Lacy J. Decentralized Trust Management[C]. Proc. of the 17th Symposium on Security and Privacy, 1996: 164-173.

2 Blaze M, Feigenbaum J, Strauss M. Compliance Checking in the PolicyMaker Trust Management System[C]. Proc. of the Financial Cryptography'98, 1998: 254-274.

3 Weeks S. Understanding Trust Management Systems[Z]. http://citeseer.ist.psu.edu/weeks01.understanding.html.

4 徐 锋, 吕 建. Web 安全中的信任管理研究与进展[J]. 软件学报, 2002, 11(13): 2057-2064.

5 张选平, 雷咏梅. 数据结构[M]. 西安: 西安电子科技大学出版社, 2002.

(上接第 174 页)

$u(x) = x^7 + x^4 + x^3 + x^2 + 1$； $u(x) = x^7 + x^6 + x^3 + x^2 + 1$；

$u(x) = x^7 + x^6 + x^4 + x^3 + 1$； $u(x) = x^7 + x^6 + x^5 + x^2 + x$；

$u(x) = x^7 + x^6 + x^4 + x^3 + x^2$；

4 结束语

本文通过这种方法构造出了密码性能与 AES 的 S 盒相当的 8×8 的 S 盒。由于在构造过程中都使用了求逆变换，因此构造出来的一批 S 盒的非线性度都为 112。无论仿射变换怎样改变都不影响 S 盒的非线性度。同样，实验表明，用构造 AES S 盒的方法所构造出来的其他 S 盒的差分均匀度也都是 4/256。惟一有变化的是 S 盒的输出比特的雪崩概率。

参考文献

1 冯登国, 吴文玲. 分组密码的设计与分析[M]. 北京: 清华大学出版社, 2000.

2 刘晓晨, 冯登国. 满足若干密码学性质的 S-盒的构造[J]. 软件学报, 2000, 11(10): 1299-1302.

3 谷大武, 徐胜波. 高级加密标准(AES)算法——Rijndael 的设计[M]. 北京: 清华大学出版社, 2003.

4 Daemen J, Rijmen V. AES Proposal: Rijndael: Version 2[EB/OL]. http://www.nist.gov/aes, 1999-09-03.

5 陈 勤, 周 律. Rijndael 分组密码与差分攻击[J]. 小型微型计算机系统, 2003, 24(4): 676-679.

6 师 军, 张福泰, 王耀燕. 高级加密标准 Rijndael 算法中的 S 盒及其实现[J]. 小型微型计算机系统, 2003, 24(7): 1207-1209.

7 张玉安, 冯登国. RIJNDAEL 算法 S 盒的等价生成[J]. 计算机学报, 2004, 27(12): 1593-1600.

8 阮传概, 孙 伟. 近世代数及其应用[M]. 北京: 北京邮电大学出版社, 2001.