AES—Rijndael 算法综述*)

Overview on AES—Rijndael Algorithm

杨景辉　王丽娜　于　戈

（东北大学信息科学与工程学院软件研究所　沈阳110004）

Abstract　AES—Rijndael algorithm is Advanced Encryption Standard which will be published by NIST of USA. It has good property. In this paper, Rijndael algorithm is introduced including its mathematical preliminaries, main characteristics, one round transformation, key expansion. Design motivation and procedure of Rijndael algorithm are discussed. Rijndael algorithm can resist common attacks such as differential cryptanalysis, linear cryptanalysis, truncated differential, square attack, related-key attack etc. At last the algorithm's limitation is given.

Keywords　AES, Rijndael algorithm, S-box, Round transformation, Key expansion

今年夏天，美国 NIST 将颁布新的数据加密标准，比利时的 Joan Daemen 和 Vincent Rijmen 提交的 Rijndael 算法作为 AES(Advanced Encryption Standard) 将被美国政府和团体广泛用作为2001年以后的加密敏感性信息的标准。

其实，早在1997年 NIST 就公开征求 AES 作为2001年以后的数据加密标准，同时提出了对 AES 的几点要求，此项举措得到世界密码界的积极响应。AES 征集通告发出之后，许多国家、企业和个人提出了自己的方案。1998年8月，AES 召开第一次候选会，确定15个算法入围；1999年3月，AES 召开第二次候选会，有5个算法入围；2000年10月，NIST 选出 Rijndael 作为 AES，可以说 Rijndael 算法代表国际上分组加密算法的最高水平。2001年2月28日颁布 FIPS-AES 草案，2月29日起，NIST 开始为期90天的公众评论，正式的 AES 将在今年夏天颁布。

本文在介绍 Rijndael 算法的同时，就其设计要点、抵抗常见攻击及实施性能方面进行一定的分析，力求对 AES—Rijndael 算法进行一个全面的介绍。

1　Rijndael 算法

1.1　数学基础

在介绍算法之前，先说明一下所用到的数学知识。在伽罗瓦域 GF(2^n)上定义了“+”、“·”、“·X ”以及带有系数的多项式的乘。对字节 a，用多项式表示为：

$a_7x^7+a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x^1+a_0$

“+”运算相当于字节的每一位进行简单的异或；同时选择一个不可约多项式：$m(x)=x^8+x^4+x^3+x+1$，两多项式相乘然后进行模多项式 m(x)的运算，就是“·”操作，因此 a(x)b(x)mod m(x)=1，求得：$a^{-1}(x)=b(x)\bmod m(x)$。GF(2^n)上求乘法逆元素就可以通过以上步骤完成，在 S-盒非线性变换中用到了这一点。在列混合变换中，将用到带有系数的多项式的乘，比较简单，在介绍列混合变换时再讨论。

1.2　Rijndael 主要特征

(1)块长可分别为128bit、192bit、256bit，密钥也可分别为128bit、192bit、256bit；

(2)Rijndael 不具有 Feistel 网络结构，其设计策略是宽轨迹策略，这就要求算法的每一步是可逆的，事实上 Rijndael 算法每一步都是可逆的；

(3)该算法把每一块表示成以字节为单位的二维数组，此数组有四行，同时也可认为是4字节为单位的一维数组；

(4)Rijndael 算法定义的所有运算都是在伽罗瓦域 GF(2^n)上进行的；

(5) 该算法具有多轮相同的运算，每一轮包括非线性替代(S-box)、行循环左移(Shiftrow)、列混合变换(Mixcolumn)以及与扩展密钥相异或；

(6)每一轮子密钥取之于扩展密钥；

(7)密钥扩展同时应用了非线性变换和循环左移。

1.3　一轮构架图

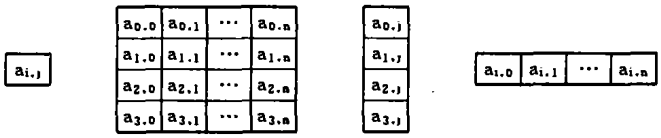

图1　加密数组块表示图

加密块数组中的 n 可以是3，5，7，所代表的加密块分别表示128bit，192bit，256bit，row 表示一行，column 表示一列，下面用 N_r、N_k 和 N_b 分别表示轮数、密钥长度(32的倍数)和块长度(32的倍数)。图1给出了一轮经历的四步运算。

图2表示的是每一轮所要经历的运算，整个算法要求有多轮这样的操作，图3是对不同的块长和密钥要求的轮数。最后一轮跟前面有所不同，最后一轮缺少列混合运算(Mixcolumn)，但这既不能提高也不能降低其安全性，这和 DES 最后一圈缺乏交换是一样的。算法的开始是密钥扩展，这是一个很有特色的运算，下面介绍 Rijndael 的密钥扩展。

```
Round(State,RoundKey)
{
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State,RoundKey);
}
```

*)本文得到教育部跨世纪优秀人才基金和高等学校优秀青年教师教学和科研奖励基金资助。

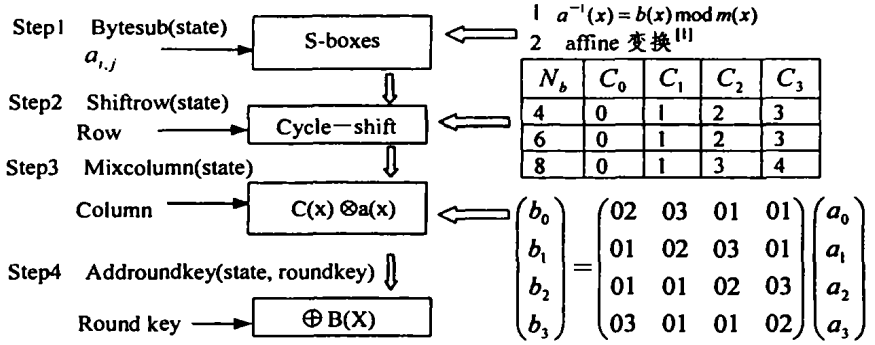

图2 一轮构架图

N_r	$N_b=4$	$N_b=6$	$N_b=8$
$N_k=4$	10	12	14
$N_k=6$	12	12	14
$N_k=8$	14	14	14

图3 轮数

1.4 密钥扩展

当 $N_K \leqslant 6$，KeyExpansion(byte key [4 * N_K], word w [Nb * (Nr+1)])

```
{ for(i=0; i<NK; i++) w[i]=(key[4*i], key[4*i+1],
    key=[4*i+2], key[4*i+3]);
    for(i= NK; 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;
      }
}
```

扩展密钥是一个以4字节大小为单位(用 Word 表示)的线性数组。当 N_k 大于6的时候有一点不同[1]。可以看出，开始 N_k 个 Word 是加密用的密钥，以后每个 Word [i]是 Word [i−1]和 Word [i− N_K]相异或，当 i 能被 N_K 整除时，先经过一定的变换，这个变换是经过一个循环左移(Rotbyte)，然后是查非线性变换表[2]，之后再与 Rcon [i/ N_K]相异或[1]。因此，就可以从扩展密钥中选择 Round key，如图4所示。

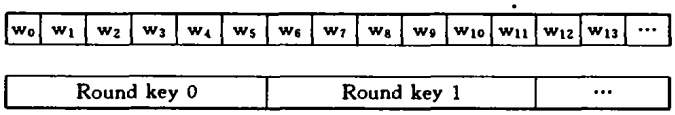

图4 密钥选择图 $N_b=6$ $N_k=4$

2 设计要点

通观整个 Rijndael 算法，包括线性和非线性运算，所有的运算定义在伽罗瓦 GF(2^n)上；执行该算法时除了要查一个查找表之外，其它的运算都是简单的异或和移位，运行的速度很快。既在每一轮运算又在密钥扩展应用了非线性操作，大大增强了抗差分和线性密码分析能力。S-盒是 Rijndael 的中心，由16个8 * 8的 S-盒并置而成，DES 有8个6 * 4的 S-盒，S-盒越大越难用差分和线性密码分析找出有用统计量，密钥扩展也是一个非常有特色的设计，相对 DES 简单的移位和压缩变换要有效得多。

2.1 轮运算要点

(1)所有操作都是可逆的；

(2)每一步对一定攻击都有很强的抵抗能力，S-盒能抵抗差分和线性密码分析，行循环左移能抵抗截断差分密码分析及 Square 攻击，列混合达到了扩散的目的；

(3)能在广泛的硬件平台上高效快速地执行；

(4)每一步操作都比较容易描述。

2.2 密钥扩展

(1)非线性操作防止了仅仅由密钥的差分而导致每一轮子密钥的差分；

(2)密钥扩展产生的效果使每一轮子密钥都不同；

(3)有效地抵抗密钥相关分析及已知部分密钥的分析[3]；

(4)消除了每轮中以及轮与轮之间的对称性；

(5)运行速度快，容易描述。

另外，在轮数选择方面，也采取比较保守的态度，用尽量多的轮运算来达到全扩散和消除各种攻击的目的。所有的一切，都使得一些攻击方法不可能对 Rijndael 有效。

3 抵抗常见攻击

(1)差分和线性密码分析：虽然这两种密码分析方法是针对 DES 提出的，从上面的介绍可知，Rijndael 在设计过程中都是以这两种密码分析方法为主要考虑对象的，非线性变换是核心；

(2)截断差分分析：对于 Rijndael，运算是以字节为基本单位而不是位，研究表明，如果 Rijndael 进行6轮或者更多轮，截断差分分析没有穷举密钥来得快；

(3)Square 攻击：Square 攻击利用面向字节的结构，是一种选择明文的攻击，研究表明，在7轮或者更多轮，这种攻击也是无效的；

(4)相关密钥分析：对于这种攻击，密码分析员选择不同的密钥进行操作，但 Rijndael 具有高度扩散和非线性操作，所以这种攻击也是不可能的；

(5)弱密钥：对于 Rijndael，不像 IDEA 具有弱密钥，对密钥的运算包括移位和非线性变换，所以对密钥的选择没有限制；

(6)密钥穷举攻击：这是对 Rijndael 最有效的一种攻击方法，如果对128位、192位和256位进行穷举的话，平均需要2^{127}、2^{191}和2^{255}次的 Rijndael 运算。这对于二十年内的计算能力是不可能的。

另外，在实施方面 Rijndael 也具有很优越的性能，Rijndael 能快速有效地在 Pentium(pro)上执行，也能通过少量代码、很少的 RAM 和小数量的循环在智能卡上执行；而且该算法能并行地执行，对未来处理器和现有的处理器都具有良好的性能，而且对处理器的体系结构没有过高的要求。

(下转第34页)

法名、类名、返回值类型、访问控制符、修饰符、参数、代码和注释字段组成。

3.4 模型数据库 MDB

模型数据库存放类图模型和抽象逻辑图过程模型的建模数据。类图模型由类图表、类图要素表和要素关系表来描述。抽象逻辑图过程模型由节点信息表描述。类图表由类图说明和类图的框架代码两个字段组成。类图要素表由要素标识、要素名、位置坐标、要素类型四个字段组成。要素关系表由关系标识、关系类型、起始要素标识、结束要素标识、方向标识五个字段组成。抽象逻辑图节点信息表由节点标识、类名、方法标识、概念类型、概念语义、逻辑类型、逻辑语义、表达式和备注字段组成。

4 用户界面

我们从规范性、易操作性、健状性方面的要求设计了如图2所示的集成开发环境用户界面。集成开发环境的用户界面包括菜单栏、快捷工具栏、资源管理工作区、建模要素符号栏、图形编辑工作区和状态栏六个部分。

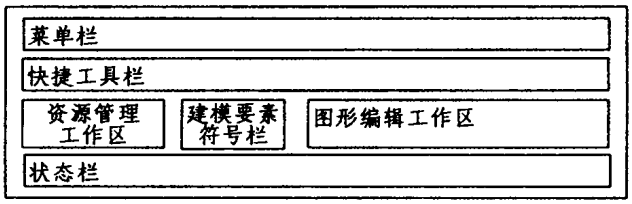

图2 集成开发环境用户界面

菜单栏设置了文件、编辑、视图、工具及帮助五个一级菜单。文件菜单有新建、打开存储、存储为、导出类型、打印及退出。编辑菜单包括拷贝、剪切、粘贴、删除、选择、刷新。在视图菜单下可对集成界面的显示内容和方式进行动态配置。工具菜单包括信息编辑、图形建模、代码生成、文档生成和选项。选项内容为路径选择、代码生成选择、项目选择的内容。在快捷工具栏上放置了一些基本操作的工具图标，包括新建、打开、保存、打印、复制、剪切、粘贴、刷新、代码、文档。在资源管理工作区中采用树形结构对系统类库和用户类库中的设计资源进行展示。在建模要素符号栏中放入类图或抽象逻辑图的建模要素，它根据不同的编辑内容自动调整建模符号。在图形编辑工作区中可对类图和抽象逻辑图进行编辑，并根据不同的编辑内容自动调入相应的图形编辑工具。对类图建模，载入类图编辑器；对类方法的过程建模则载入抽象逻辑图编辑器。抽象逻辑图编辑器将编辑内容分为概念、逻辑和实现三层视图，每一层视图都以树形方式组织，可同时对三层视图或单独对三层视图中的任一层进行编辑。编辑方式可在视图菜单中进行配置。系统提供的刷新操作可保证资源管理工作区的显示内容与图形编辑工作区的类图模型保持一致。

为方便使用，对资源管理工作区和图形编辑工作区中的所有编辑元素都提供右键快捷菜单，并且实现了选择对象与可执行操作的动态关联，对不可用的操作选项自动进行屏蔽。

结束语 我们以 UML 类图和抽象逻辑结构图为基础设计的 JVOOPSS 系统支持从类图建模、类方法的过程建模到完整 JAVA 程序代码生成的正向工程，实现了 UML 类图与抽象逻辑结构图的有机结合，实现了可视化建模与编码的统一，可用于 JAVA 应用程序和 JAVA 小程序的可视化开发。我们的系统与其它同类系统相比具有以下特点：

1)用抽象逻辑图填补了 UML 模型中缺乏的过程静态建模部分，将静态建模的级别细化到类方法一级，使系统的静态建模机制较为完整，因而系统能够根据设计模型生成完整的 JAVA 程序源代码。其它系统大多只能根据类图模型生成类的框架代码，对类的方法代码主要靠设计者使用传统方法进行设计与编码。

2)系统支持从类体系结构设计到类方法详细设计的 JAVA 面向对象可视化程序设计全过程，支持从类图建模、类的定义、类方法过程建模到 JAVA 程序源代码生成的平滑过渡。

3)系统以支持 JAVA 可视化程序设计为目标，不涉及复杂的动态建模部分，因而简洁实用，操作简单。

我们已成功应用 JVOOPSS 开发出能在网上运行的 JAVA 游戏程序，证实了系统的可用性。实践表明该系统对提高 JAVA 程序设计的效率，明显改善程序的可理解性和可维护性是有效的。

参考文献

1 Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999
2 Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Addison-Wesley, 1999
3 刘建宾，龚世生. 抽象逻辑结构图及其应用. 计算机科学，1996，23(6)：83～86
4 刘建宾. JAVA 过程蓝图. 计算机科学，2000，27(7)：87～91
5 刘建宾，郝克刚，龚世生. 抽象概念结构图到 JAVA 过程蓝图的平滑过渡及一致性. 计算机科学，2001，28(8)

(上接第49页)

结论 NIST 之所以选择了 Rijndael 算法，是由于 Rijndael 是集高安全性、高性能、高效率、易实现及伸缩性强于一身的[4]。特别地，Rijndael 能通过广泛的计算环境，不管是用反馈模式还是非反馈模式，使得硬件和软件性能达到最优。密钥建立高效且灵活。Rijndael 很低的存储要求，使之能在有空间限制的环境中执行，Rijndael 能很好地抵抗各种攻击。另外，密钥和块长都设计得很灵活，同时算法在轮数选择方面能提供一定的灵活性，最后，Rijndael 轮运算对指令级并行很具潜力。

但是，加密的逆运算有一些限制。在智能卡上解密过程相对加密过程难于执行，需要更多的代码和循环。对于软件，加密和解密需要不同的代码和查找表。对于用硬件来实施加密和解密，解密只能部分地重用加密过程用到的电路。

参考文献

1 Daemen J, Rijmen V. The Rijndael Block Cipher Version 2. 03/09/99
2 American NIST, Draft of FIFS-AES. 02/28/01
3 冯登国，裴定一. 密码学导引. 科学出版社，1999
4 http://www.nist.gov/aes