文章编号:1000 - 582X(2005)10 - 0079 - 03

结合混沌实现 Rijndael 算法的密钥调度*

张文忠,黄席樾,陈善勇
(重庆大学 自动化学院,重庆 400030)

摘 要:加密是保护个人和企事业单位重要信息乃至国家机密的重要手段,Rijndael 加密算法作为新的高级加密标准,由于其诸多优点而具有广泛的应用前景.现代的加密算法都是基于密钥的,然而,迄今为止,很少有人研究 Rijndael 算法的密钥调度部分,尤其是如何生成 Rijndael 算法初始密钥.针对这一情况,研究了混沌 Logistic 映射,利用 Logistic 映射产生的混沌序列的高随机性特点,采用混沌算法来生成初始密钥,并给出了相应的 CPLD 实现.

关键词:Rijndael;密钥调度;CPLD;混沌
中图分类号:TK14 文献标识码:A

信息安全关系着企事业单位的合法权益乃至国家的经济、政治、军事等方面的安全问题,所以,如何保障信息安全具有十分重大的意义.

加密技术是一种有效的保护信息安全的手段. Rijndael 算法作为新的高级加密标准(AES),它是比利时 2 位密码专家 Joan Danmen 和 Vincent Rijmen 开发的,其设计思想新颖、设计原则明确、数学基础好,由于其具有安全性好、性能稳定、加解密速度快、效率高、易于软硬件实现、内存需求小等优点而得到了人们的广泛关注.现在 Rijndael 算法已应用于软件产品保护和信息安全等领域.但是,在已有的 Rijndael 算法研究及实现方面,很少有人研究 Rijndael 算法密钥调度部分的初始密钥生成这一重要部分.众所周知,现代加密技术的安全性都是基于密钥的,若密钥被破解,则系统也就无安全可言了.基于这一情况,笔者重点研究了混沌 Logistic 映射,分析了 Rijndael 算法的密钥调度部分,提出了把 Logistic 映射应用于 Rijndael 算法初始密钥生成的新思想、新方法,并利用 CPLD 进行硬件实现密钥调度部分,从而提高了 Rijndael 算法的安全性.

1 混沌 Logistic 映射

混沌系统是一种非线性的高度复杂的动态系统,具有对初始条件极为敏感的特征,并且由其所生成的混沌序列具有伪随机的特性以及遍历性、良好的统计性等.混沌的一个最基本的特征就是对初始条件的敏感性.任意 2 个有细小差别的初始值 x_0 和 x'_0,在经过多轮迭代后,得到的两个迭代序列 $\{x_0, x_1, \cdots, x_n\}$ 和 $\{x'_0, x'_1, \cdots, x'_n\}$ 会发生背离,使输出的结果完全不相关.利用这一特性,可以产生许多非相关、高随机性的混沌序列.

在这里,笔者选取混沌 Logistic 映射[1].离散时间动态系统 Logistic 映射定义如下:

$$x_{n+1} = \mu * x_n * (1 - x_n),$$
$$0 < x_n < 1, \quad 0 < \mu < 4,$$

其中 x_n 为状态,μ 为参数.以初始值 x_0 为初始状态进行迭代,再通过调节参数 μ,就可以得到一个理想的混沌序列. Schuster 导出当 $\mu_\infty < \mu < 4$,$\mu_\infty = 3.569\,945\,6$ 时,Logistic 映射处于混沌状态,其周期 $N \to \infty$.图 1 所示是当 $\mu = 3.0 < \mu_\infty$,$x_0 = 0.625$ 时的混沌序列,序列在迭带一定次数后,会发生收敛现象,出现周期性.图 2 所示是 $\mu = 4.0$,$x_0 = 0.625$ 时迭代产生的混沌序列,所产生的混沌序列具有很好的随机性,近似于现实世界的白噪声随机序列.

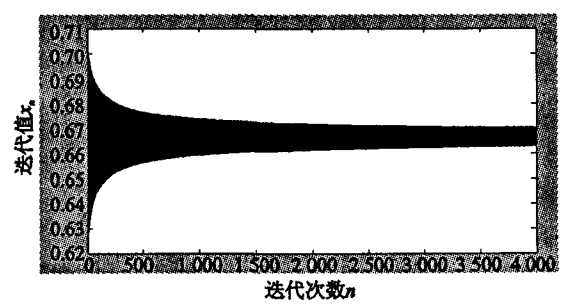

图 1 $\mu = 3.0$,$x_0 = 0.625$ 时的随机序列

* 收稿日期:2005 - 05 - 10
基金项目:国家自然科学基金资助项目(69674012)
作者简介:张文忠(1978 -),男,河南辉县人,重庆大学硕士,主要从事加密方法的研究和实现及智能机器人控制.

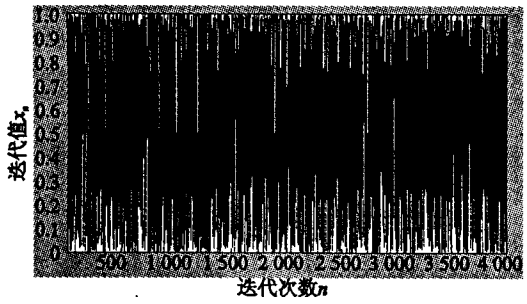

图2 $\mu=4.0, x_0=0.625$ 时的随机序列

2 Rijndael 算法的密钥调度

Rijndael 算法[2]是一种迭代型分组加密算法,其明文分组长度和密钥长度可变,明文分组长和密钥长度可各自独立指定为 128、192、256 位. Rijndael 算法把每 8 位二进制数作为一字节,以字节为最基本的处理单位,将输入、输出及中间结果称作状态,其加密过程由密钥扩展、初始密钥加操作以及轮变换组成. 下面只分析 Rijndael 算法的密钥调度部分.

密钥调度由 3 个模块组成:初始密钥生成、密钥扩展和轮密钥存储.

2.1 初始密钥生成

现代密码学中,算法或密码系统的安全性是基于密钥的,所以,如何生成初始密钥是很关键的. 笔者采用混沌算法产生初始密钥. 由于混沌系统产生的混沌现象具有非周期、不收敛、有界、对初始状态和参数的敏感性等特点,所生成的混沌序列有着良好的高随机性,因此,完全可以利用混沌算法提供密钥.

2.2 密钥扩展

按一定算法,将初始密钥进行扩展,生成加密密钥,即扩展密钥[3],整个扩展密钥可看成是一个元素为字数据的一维数组,数组大小为 $Nb*(Nr+1)$,其中,Nb 为状态的列数,Nr 是加密轮数. 扩展算法[4]可示意为:

```
KEp (wd KInl[Nk],wd KEp[Nb*(Nr+1)])
{
  for(i=0;i<Nk;i++)
   KEp[i]=KInl[i];
  for(i=Nk;i<Nb*(Nr+1);i++)
  {
   wd tp=KEp[i-1];
   if(i%Nk == 0)
     tp=SB (RB (tp))^Rc [i/Nk];
   KEp[i]=KEp[i-Nk]^tp;
  }
}
```

SB 为字节替换,有以下 2 步:

1) 在有限域 GF(2^8)上,求每一字节的逆元,"00"的逆元为其自身;

2) 进行仿射变换.

RB 是以字节为单位循环移位,Rc[j]为轮常数.

2.3 轮密钥存储

将扩展密钥以轮密钥的形式存储.

3 密钥调度模块 CPLD 实现

硬件实现原则为:面积要小、速度要快、成本要低、容易实现. 面积小也就意味着尽量减少所用的芯片逻辑单元数目,速度快指的是单位时间内处理的数据要尽量多,通常速度越快,占用的逻辑单元越多,这两者是相互矛盾的,在设计时要综合考虑,根据实际需要作出合理的决定. 本次设计中,面积小是主要的原则;易实现则是指易用 AHDL 描述,对应的硬件电路要尽可能简单.

3.1 初始密钥生成

由上可知,对于 Logistic 映射,当 $3.569\,945\,6<c<4$ 时,Logistic 映射处于混沌状态,即初值 x_0 或参数发生微小的变化,经过一段时间后,生成的序列 $\{x_n^*\}$ 便与先前的序列 $\{x_n\}$ 相互背离,即它们没有了相关性,由 $\{x_n\}$ 很难推出 $\{x_n^*\}$. 利用这一特性,可利用这个式子产生初始密钥,理由是:只要稍微改变一下初值 x_0 或参数 c,经过相同的时间,生成的密钥可完全不同. 如果再结合使用一定时器和一选择电路,则可更为灵活地生成密钥,从而使得破译密钥变得更加困难,极大地提高算法的安全性.

利用 CPLD 进行硬件实现时,使用了 Maxplus II 开发工具以及 Altera 公司提供的 AHDL 硬件描述语言,采用自顶向下、逐步求精的方法,完成系统描述,最后用下载电缆直接把描述装入 CPLD 相应芯片.

显然,用 CPLD 实现 Logistic 映射非常简单,只需要设计乘法电路和减法电路. 初始密钥生成模块的硬件示意图如图 3.

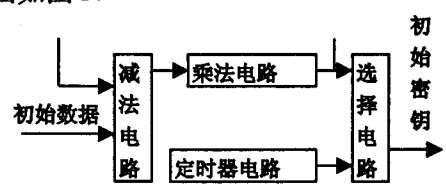

图3 初始密钥生成模块的硬件示意图

在上面的电路中,初始数据中包含了初始值 x_0 和参数 μ,减法电路实现 $(1-x_n)$,令 $x=1-x_n$,则乘法电路实现 $\mu*x*x_n$,为减少芯片面积,采用反馈电路实现 Logistic 映射,定时器电路在芯片内部实现. 此处的选择电路不是通常意义上的选择电路,它的含义是:电路根据定时器的时间值来选择 Logistic 映射产生的混沌序列中的某一数据. 这样,既可以通过改变初始值 x_0 和参数 μ 来使初始密钥发生变化,也可以通过调节定时器的定时值改变初始密钥,当然,还可以通过同时调节初始值 x_0、参数 μ 和定时器定时值来设置初始密

钥. 所以,初始密钥的产生具有很大的灵活性与可控性. 另一方面,定时器只在芯片内部实现,外部没有引脚,它隔离了初始密钥与初始数据的关系,进一步提高了系统的安全性.

3.2 密钥扩展

分析密钥扩展算法可知,整个密钥可看成一数组,密钥扩展就是用于产生 $Nk \sim Nb*(Nr+1)$ 之间的数组元素,若以字为单位扩展密钥,每字有 32 位,则当密钥长度为 128 位时,每组密钥有 4 个字,下一密钥组与前一密钥组的关系用式子[5]表示为: $Kn_0 = (\mathrm{SubByte}(\mathrm{RotByte}(Kp_3))\hat{}\mathrm{Rcon}[i/4])\hat{}Kp_0$, $Kn_1 = Kn_0\hat{}Kp_1$, $Kn_2 = Kn_1\hat{}Kp_2$, $Kn_3 = Kn_2\hat{}Kp_3$. 在上面的公式中,$Kp_0 \sim Kp_3$ 可看成是变量,表示已扩展好的字密钥,$Kn_0 \sim Kn_3$ 也可看作是 4 个变量,表示待扩展的下一字密钥. 利用上式可方便地用 CPLD 实现算法. 在硬件实现时,对于 SubByte 变换,其求字节逆元及仿射变换均为一一对应运算,所以,SubByte 变换的输入和输出也是一一对应关系,可用一查找表(即 ROM)实现字节变换;而对 RotByte 操作,用一简单的移位电路便可完成其功能;模 2 加部分用一异或电路可方便地实现. 密钥扩展模块[6]的图形表示为:

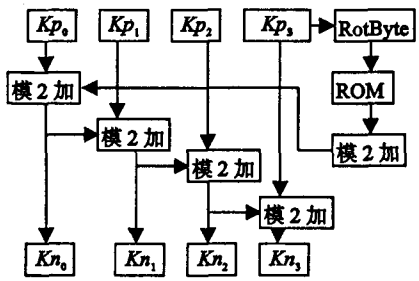

图 4　密钥扩展模块

为减少逻辑单元数,用反馈电路实现密钥扩展,即 $Kn_0 \sim Kn_3$ 作为下一级的已扩展好的字密钥 $Kp_0 \sim Kp_3$ 再扩展密钥,直到完成密钥扩展.

3.3 轮密钥存储

显而易见,利用一存储器可实现轮密钥存储.

4 结　语

CPLD 实现 Rijndael 算法,集灵活性、安全性和易实现性于一体,具有安全性高、效率高、性能良好、易开发、成本低等优点. 由于现代的加密算法的安全性都是基于密钥的安全性,因此,如何设置、生成初始密钥便具有重大意义. 仿真结果表明: Logistic 映射产生的混沌序列具有很好的随机性. 这样,便更好地保证了 Rijndael 算法的安全性.

参考文献:

[1] 赵嘉莉,罗四维,温津伟. 基于神经网络的混沌加密算法[J]. 计算机研究与发展,2001,38(12):1 475 - 1 479.

[2] 岳轩,陈维海. Rijndael 加密算法及其实现[J]. 河北省科学院学院, 2002, 19(14):213 - 236.

[3] CRINTANCHITU, DAVIDCHIEN, CHA-RLES CHIEN, et al. A Hardware Implementation in FPGA of the Rijndael Algorithm[Z]. The 2002 45th Midwest Symposium on Circuits and Systems, Oklahoma: Tulsa, 2002.

[4] ANDERSON CATTELAN, ZIGIOTTO, ROBERTO. A Low-cost FPGA Implementation of the Advanced Encryption Standard Algorithm[Z]. Proceedings of 15th Symposium on Integrated Circuits and Systems Design, Brazil: Porto Alegre, 2002.

[5] ULRICHMAYER, CHRISTOPHER, OELSNER, et al. Evaluation of Different Rijndael Implementations for High End Servers[Z]. IEEE International Symposium on Circuits and Systems, Arizona: Scottsdale, 2002.

[6] ALEX PANATO, MARCHELO BARCE-LOS, RICARDO REIS. An IP of an Advanced Encryption Standard for Altera Devices[Z]. Proceedings of 15th Symposium on Integrated Circuits and Systems Design, Porto Brazil: Alegre, 2002.

Realizing Key Schedule of Rijndael Algorithm Integrating with Chaos

ZHANG Wen-zhong, HUANG Xi-yue, CHEN Shan-yong

(College of Automatization, Chongqing University, Chongqing 400030, China)

Abstract: Encryption is an important method which is used to protect the information of an individual and enterprises and the secret of nations. Rijndael algorithm is the new advanced encryption standard, and has extensive application foreground because of its much merit. Modern encryption algorithms are based on keys. However, so far few of people research into the key schedule of rijndael algorithm and especially do research on the generation of its initial keys. This paper researches into the mapping of Logistic, makes use of a chaos algorithm to produce initial encryption keys because it can create chaos sequences of high randomicity. Further more, corresponding CPLD implementation is given.

Key words: Rijndael; key schedule; CPLD; chaos

(编辑　张　苹)