一种基于动态S-盒P-盒的快速分组密码算法——DSP

陈利科[1] 张润彤[1,2]

(大连海事大学信息科学技术学院 大连 116029)[1] (北京交通大学经济管理学院 北京 100044)[2]

摘 要 密钥相关加密结构作为一种较安全的密码结构受到密码工作者的广泛关注,然而现有该类算法的安全缺陷和十分复杂的算法初始化过程,严重地限制了算法的使用。因此,提出一种基于密钥相关Feistel结构的快速分组加密算法,算法通过结合密钥相关的动态S-盒和密钥相关动态P-盒两种基本密码组件,设计一种更加安全的Feistel轮加密结构,可以使算法在较少的轮数内达到安全。同时,该算法通过采用快速置乱算法生成S-盒、P-盒,改进了现有该类算法子密钥生成算法效率极低的缺点。为了得到更好的兼容性,算法仅选用基于字节的密码操作,使得算法广泛适用现有的大多数处理器。算法的最大特点就是使用了密钥相关的动态S-盒(DS)和动态P-盒(DP),因此该密码结构命名为DSP结构,该算法为DSP分组密码算法。分别用C和Java在不同Pentium PC上实现了该算法;实验结果表明,该算法有着较好的加密解密效率,以及相对快速的算法初始化过程。

关键词 DSP结构,动态S-盒,动态P-盒,分组密码算法

Novel Software Block Cipher Using Dynamic S-box and P-box

CHEN Li-ke[1] ZHANG Run-tong[1,2]

(College of Infomation Science and Technology, Dalian Maritime University, Dalian 116029, China)[1]

(School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China)[2]

Abstract Block ciphers based on key-dependent cipher structures have been investigated for years, however, their overall performance in terms of security and speed has not been sufficiently addressed. We proposed a 128-bit Feistel block cipher, which engages both dynamic S-box and dynamic P-boxes that are all key-depen-dent. With these two key-dependent transformations, the internal structure of this cipher algorithm was secured, so as to resist the linear and differential cryptanalysis in a few round encryptions. Hence, the encryption and decryption functions are very efficient. We named this key-dependent structure the DSP structure, and the cipher DSP. A fast permutation algorithm was used to generate both the dynamic S-box and dynamic P-boxes. This greatly compensates the performance penalty of complex key schedule. The basic operations selected in DSP are all efficient bytewise operations, so the algorithm will have a reasonable fast speed on recent processors, 16-bit processors and smart cards as well as 8-bit processors. We implemented the algorithm with C and Java on two different PCs with Pentium processors, and estimated the optimized assembly performance. The experimental results and the estimation show that DSP has a very fast encryption/decryption speed and a reasonable fast key scheduling implementation.

Keywords DSP structure, Dynamic S-box, Dynamic P-box, Block cipher

1 引言

自差分密码分析[1]和线性密码分析[2]被提出以来,如何使分组密码算法在更少的轮数内有效地抵抗这两种攻击成为了现代分组密码研究的焦点问题。

"差分"和"线性"之后的密码算法都采用不同的方式来抵抗这两种攻击,它们的核心思想大致可以分为两类:第一类算法以Square[3],AES[4],MISTY[5],Camellia[6]和Twofish[7]等算法为代表,它们采用最大差分传播概率和最大线性相关性概率都尽可能小的固定(密钥无关)密码结构来实现分组密码算法。在这一原则的指导下,算法必须谨慎地选择特殊的S-盒操作以及与其密切相关的线性混合操作。而且作为现代密码算法中唯一的非线性操作,S-盒不光要有尽可能小的差分传播概率,其输入输出最大的线性相关性也必须尽可能低;而与S-盒相关的线性混合层要尽快地把S-盒的非线性性传播到整个分组,以实现雪崩效应。这类算法的最大优点是它们抵抗差分密码分析和线性密码分析的能力可以得到证明;但是,通常它们都要执行较多的轮数来达到安全,而且固定的密码结构也为潜在的攻击提供了可能。

另一类算法采用与密钥相关的密码结构,来增强密码算法的安全强度,如密钥相关的动态S-盒。由于无论是差分密码分析还是线性密码分析,都需要对S-盒的性质进行具体的

到稿日期:2008-04-01 本文受国家自然资金项目(60773033)资助。

陈利科(1979—),男,博士生,研究方向为信息安全、通讯技术等,E-mail:chenlike8888@sina.com;张润彤(1963—),男,博士后,教授,博士生导师,研究方向为电子商务、知识管理、智能控制等。

分析，因此该类算法可以通过隐藏 S-盒的特性来抵抗线性密码分析和差分密码分析。理论上，这类算法比第一类算法的安全强度高，执行较少的轮数就可以达到安全，所以此类算法应该得到十分广泛的应用。然而，此类算法受到的重视远远不如第一类算法，这主要是由于现有的该类算法，如 Khufu[8] 和 Blowfish[9]，都存在着一些安全性和实用性的问题。

Khufu 和 Khafre[10] 都是分组长度为 64、密钥长度为 512 的快速软件密码算法。对算法的分析表明，Khufu 比 Khafre 执行的轮数少，但前者比后者更安全[10]。这主要是因为 Khufu 使用的是一个密钥相关 8-32-bit 的 S-盒，而 Khafre 使用的是一个相同大小的固定 S-盒。Khufu 具有十分高效的加密解密函数，但是其密钥扩展过程过于复杂，这就大大限制了该算法的实用性。而且算法也存在一定的安全性问题，Khufu 使用的是 8-32-bit 的放大型 S-盒（n-bit 输入，m-bit 输出，$n<<m$），研究表明，放大型 S-盒容易产生不可能差分特征（impossible differentials）；文献[10]中提供的一些差分特征，还可以使雪崩效应推迟 8 轮。

Blowfish 是另一个典型的采用密钥相关密码结构的密码算法，其分组长度为 64-bit，密钥长度可变，其安全强度主要来自 4 个 8-32-bit 密钥相关 S-盒。由于该算法中仅包含了一些最简单的基本操作，因此算法有着十分高效的加密解密速度。可是，该算法的密钥扩展函数十分复杂，连设计者本人都认为该算法所用的密钥不适合经常改变，宜提前计算子密钥和 4 个动态 S-盒，并将其存储以备使用。而且 Blowfish 的 S-盒完全随机生成，算法未作任何限制，因此易产生弱密钥，使 S-盒发生碰撞[11]。

为了进一步研究密钥相关密码结构，解决现有算法存在的问题，本文提出了一种新的基于密钥相关 S-盒和密钥相关 P-盒的分组密码算法。该算法采用 Feistel 全局结构，支持可变长密钥，对 128 比特分组数据进行加密解密。算法通过采用一种快速置乱算法来提高密钥扩展的效率，生成一个 8-8-bit 无碰撞密钥相关的动态 S-盒（key-Dependent S-box，DS）和 r 个作用于 8 个位置元素的动态 P-盒（key-Dependent P-box，DP）（r 为算法的加密轮数），S-盒和 P-盒都具有很强的随机性。S-盒的随机性使密码分析者将无法通过对具体 S-盒的分析来攻击密码，有力地抵抗了基于具体 S-盒分析的密码攻击方式；动态 P-盒被引入与动态 S-盒相结合，有效地隐藏了算法的差分轨迹和线性轨迹，从而彻底地抵抗了差分密码分析和线性密码分析。算法的最大特点就是使用了密钥相关的动态 S-盒（DS）和动态 P-盒（DP），因此我们命名该密码结构为 DSP 结构，该算法为 DSP 算法。算法所采用的基本操作都是高效的字节操作，这为算法提供了很强的通用性，使算法在 32 位、16 位甚至 8 位处理器上都能有较高的运行效率。本文分别用 C 和 Java 实现了该算法，结果表明，本算法除了实现了高速的加密解密功能，还具有相对快速的算法初始化过程。

本文第 2 部分介绍 DSP 算法以及相应的设计原理；第 3 部分在不同环境下测试算法；最后为结论。

2 DSP 分组密码算法

2.1 DSP 算法基本思想

DSP 被定义为一个 128-bit 迭代型 Feistel 分组密码算法，算法支持可变长密码，但密钥长度通常要大于 128 比特。算法加密的轮数可根据用户的需求自行调整，要注意加密轮数必须为偶数，本文建议加密的轮数要大于等于 8。图 1 给出了算法的全局结构。

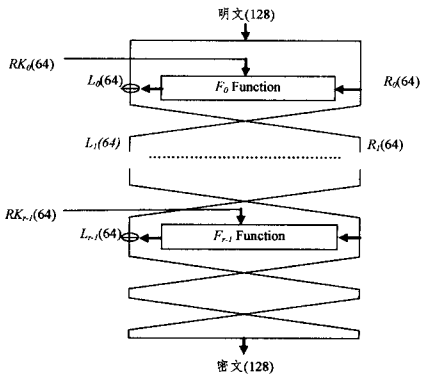


图 1 DSP 算法的全局结构

算法的核心思想就是通过使用密钥相关的动态 S-盒和动态 P-盒替换传统分组密码算法中使用的静态 S-盒和静态 P-盒。作为 Shannon 密码学理论的两种最基本密码操作，S-盒和 P-盒的设计至关重要；而本文通过快速置乱算法，生成密钥相关的动态 S-盒和 P-盒，就是要隐藏密码算法的具体内部结构，使算法更难于分析，执行较少的轮数就能达到安全。

2.2 密码算法的基本操作

现代密码算法通常选用以下 4 种基本操作来构成分组密码 SPN 的轮结构：逻辑运算、算术运算、移位操作和查表操作。

逻辑运算中的异或运算可以说是现代密码学中使用最广泛的一种基本操作，因为它在任何系统下都有最高的运行效率，该操作是密码算法理想的选择。

算术运算中的操作，如加、减、乘，在面向 32 微处理器的软件加密算法中较为常用，因为在部分处理器下，以上操作都可以在一个时钟内完成，而且操作的安全强度相对较高，特别是乘法操作；但是，16-bit 乘法在 8-bit 和 16-bit 处理器上执行效率非常低，因此选择该类操作的密码算法，缺乏较好的通用性。

移位操作，尤其是循环移位操作，也是现代密码算法中常用的一种基本操作，它能间接地改善数据的扩散效果，遗憾的是在某些处理器上（如 Pentium）循环移位的执行代价非常高（4 个时钟），而且动态循环移位不能和其他处理器指令“配对”并行运行（instruction “pairing”），阻断了流水线，大大影响算法的工作效率。

在软件密码算法中，查表操作的效率主要取决于内存的存取速度的快慢，早期的处理器对内存访问的开销要远远大于寄存器操作，然而随着处理器技术的不断发展，近期的处理器已可以在一个时钟内完成对内存的读写，这大大提高了查表操作在密码算法中的适用性。

综上，DSP 仅选用逻辑运算（异或操作）和查表操作来设计算法，以求在大多数处理器上都有较高运算效率。

2.3 密钥扩展

DSP 的密钥扩展包括 3 部分，DS 生成、SK（子密钥）扩展和 DP 生成。这一节，我们先介绍 DS 和 DP 的生成。

2.3.1 DS&DP 生成

DSP 采用著名流密码算法 RC4[12] 的初始化函数（key

scheduling algorithm)作为置乱算法，用以快速生成 DS 和 DP。该函数定义如下：

```
Function Permute(T,K,M){
        J = 0;
        For I = 0 to M-1
          J = (J+T[I]+K[ I % L]) % M;
            Swap(T[I],T[J]);
        End_For }
```

程序中 SWAP 表示信息元素位置互换，M 表示信息元素的个数，*Len* 为控制密钥的长度。由以上程序我们看出，输入信息 T 在密钥 K 的控制下进行置乱，由 I 遍历的每个元素都与其它元素或其本身进行一次位置互换，因此能够提供非常好的随机性，而且该算法的执行效率很好，十分适合用来初始化 DS 和 DP，初始化过程如下：

```
For I = 0 to 255
  S[I] = I;
End_For
DS=Permute(S,SK,256);
For I = 0 to N/2-1
    P[I] = N/2-I;
End_For
DP=Permute(P,PK,N/2);
```

这里，S 和 P 分别表示 DS 和 DP 的初始状态，该状态可根据应用的要求而定义。$N=16$ 表示分组大小(以字节为单位，8-bit×16=128-bit)，由于我们采用 Feistel 结构，因此轮函数 F 只对分组信息的一半进行处理，因此每个 DP 包含 8 个位置信息。UK 和 PK 分别为置乱控制密钥，其中 UK 被定义为用户提供的密码密钥，PK 由 UK 扩展得到，详见 2.3.2 节。因此，在控制密钥不公开的前提下，DS 和 DP 可以被安全快速地计算出来。

2.3.2 子密钥扩展

DSP 的子密钥包含两部分，前一半(RN/2 字节)作为轮密钥 RK，用来控制轮函数 F，后一半(RN/2 字节)作为 DP 生成控制密钥 PK，用来控制 R 个 DP 的生成。我们知道，密码分析者可能会通过种种手段，恢复出部分的子密钥，所以子密钥扩展算法就必须能够抵抗分析者通过部分子密钥递推出全部子密钥乃至密码密钥的攻击。本文把刚生成的 DS 作为安全变量，引入密钥扩展算法，以抵抗上述攻击。具体算法如下：

```
For I = 0 to Len-1
    TK[I] = K[I] ⊕DS[ K[I]+i ];
End_For
For I = Len to RN-1
  Tmp= TK[I - Len] ⊕TK[I-Len/2] ⊕(TK[I-1]);
  TK[I] = DS[Tmp];
End_For
```

这里"⊕"表示异或操作。通过以上算法 DSP 可以把任意长度的密钥加以扩展，但是，如前所述，我们建议密钥长度要大于 128 比特。

2.4 线性变换

线性混合层的设计应该遵循安全和快速的原则。要达到安全，就要使算法在连续两轮产生尽可能多地活动分支数，要实现快速，就应该选用并行性好、高效的处理器指令来设计算法。基于以上原因，本文将 Camellia 算法的线性变换引入 DSP，因为该算法可以完全使用异或操作来实现，使得 DSP 无论在 32-bit 处理器、16-bit 处理器、智能卡、还是 8-bit 处理器上都能实现较高的运行速度。线性变换定义如下：

DB′[0]=DB[0]⊕DB[2]⊕DB[3]⊕DB[5]⊕DB[6]⊕DB[7];

DB′[1]=DB[0]⊕DB[1]⊕DB[3]⊕DB[4]⊕DB[6]⊕DB[7];

DB′[2]=DB[0]⊕DB[1]⊕DB[2]⊕DB[4]⊕DB[5]⊕DB[7];

DB′[3]=DB[1]⊕DB[2]⊕DB[3]⊕DB[4]⊕DB[5]⊕DB[6];

DB′[4]=DB[0]⊕DB[1]⊕DB[5]⊕DB[6]⊕DB[7];

DB′[5]=DB[1]⊕DB[2]⊕DB[4]⊕DB[6]⊕DB[7];

DB′[6]=DB[2]⊕DB[3]⊕DB[4]⊕DB[5]⊕DB[7];

DB′[7]=DB[0]⊕DB[3]⊕DB[4]⊕DB[5]⊕DB[6];

这里，DB 为轮函数 F 的中间变量，DB′为线性变换后的中间变量。

2.5 轮函数 *F*

DSP 的轮函数包括 4 组连续的操作：

1) 密钥添加层：每一轮中，F 函数的 8 字节输入变量都要和 8 个字节的轮密钥相异或。其定义如下：

```
For I = 0 to N/2-1
DB[I]=DB[I] ⊕RK[I];
  End_For
```

2) DS 变换：与轮密钥相结合的 8 字节中间变量，通过表查询操作被替换为 DS 中对应的字节元素。

```
For I = 0 to N/2-1
    DB[I]=DS[DB[I]]
End_For
```

3) 线性混合层：将经过 DS 的 8 字节中间变量，按 2.3 节中算法进行变换处理。

4) DP 变换：根据 DP，将线性变换后的结果中间变量的顺序置乱，并把置乱后的结果作为轮函数 F 的输出。定义如下：

```
Tmp=DB;
For I = 0 to N/2-1
    DB[DP[I]]=Tmp[I];
End_For
```

3 仿真试验

我们分别用 C 和 Java 在两台不同性能的 PC 机上实现本文算法，并在不同实验环境下与优化后的 Blowfish 算法和 Khufu 算法进行比较。表 1 给出了具体实验环境，表 2 给出了算法实现参数的选择。

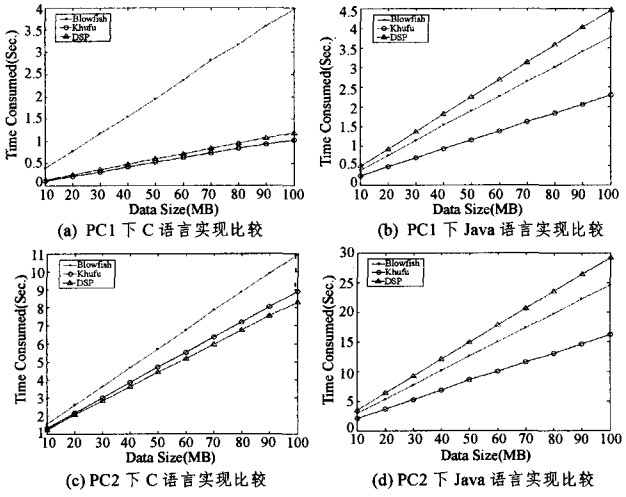
表 1 实验环境

	处理器类型	内存	操作系统
PC1	Celeron 2.4 GHz	512MB	Windows XP
PC2	Celeron 400MHz	160MB	Window 98

表 2 算法实现参数

	Blowfish	Khufu	DSP
分组大小(比特)	64	64	128
实现轮数	16	32	8

其中C程序在Visual C++ 6.0下编译，Java程序的编译器选择为JavaSoft的JDK1.5。为了提高实验的准确性，我们取10次实验的平均值作为实验结果。密码的操作模式选择为电子密码本(ECB)模式，实验通过对10～100MB随机块数据的加密来计时并计算效率，由于Feistel密码的对称性，DSP，Blowfish和Khufu加/解密速度相等，因此本文只对算法加密的效率进行比较。图2为算法在不同实验环境下的加密效率的比较。



(a) PC1下C语言实现比较　(b) PC1下Java语言实现比较

(c) PC2下C语言实现比较　(d) PC2下Java语言实现比较

图2　加密效率比较

从图2我们可以看出，在不同环境下，Blowfish算法和Khufu算法在两台PC上都有着相当高的运行效率，这是因为Blowfish和Khufu算法都是针对32位处理器设计的，当算法运行在16位处理器或者8位处理器上时，算法加密单位数据所用的时钟周期将显著升高，严重影响算法的工作效率；而且随着处理器性能的降低，Khufu算法相对于DSP的速度优势逐渐降低，在Celeron 400MHz处理器上，DSP的C语言代码加密速度已高于Khufu算法。DSP算法在各种环境下都具有相对较高的运行效率，并且该算法以字节操作为基础，在不同的处理器上具有良好的兼容性。

结束语　本文提出了一种新的128位Feistel分组密码算法——DSP。该算法将密钥相关的动态S-盒与密钥相关动态P-盒相结合，隐藏了密码算法的内部结构，从而有效地抵抗差分密码分析和线性密码分析。算法的设计仅使用了异或和查表两种基于字节的基本操作，因此算法具有良好的实现效率和兼容性。在算法的密钥初始化部分，我们采用RC4算法中的状态初始化函数作为置乱算法，实现了快速的密钥扩展，从很大程度上弥补了密钥相关结构算法初始化过程开销巨大、运行缓慢的问题。我们在Pentium处理器上分别用C和Java实现了DSP，实验结果表明，加密解密过程及密钥初始化均十分高效。

参考文献

[1] Biham E, Shamir A. Differential cryptanalysis of the Data Encryption Standard [M]. New York: Springer-Verlag, 1993

[2] Matsui M. Linear cryptanalysis method for DES cipher [C]// Advances in Cryptology -EUROCRYPT ' 93 Proceedings. Springer-Verlag, 1994: 286-397

[3] Daemen J, Knudsen L R, Rijmen V. The block cipher Square [C]// Fast software encryption-FSE'97. Haifa, Israel: Springer Verlag, January 1997: 149-165

[4] Advanced Encryption Standard [S]. FIPS-197. National Institute of Standards and Technology, Nov. 2001

[5] Matsui M. New block encryption algorithm MISTY [C] // Fast Software Encryption - 4th International Workshop (FSE'97), LNCS. vol. 1267. Springer-Verlag, 1997: 54-68

[6] Aoki K, Ichikawa T, Kanda M, et al. Camellia: A 128-bit block cipher suitable for multiple platforms-Design and analysis [C]. submitted to NESSIE. Available at: http://www.cryptonessie.org, 2000

[7] Schneier B, Kelsey J, Whiting D, et al. Twofish: A 128-Bit Block Cipher [C]// First Advanced Encryption Standard (AES) Conference. Ventura, California, USA, 1998

[8] Merkle R C. Fast software encryption functions [C]// Proc. CRYPTO'90. LNCS. vol. 537. Springer-Verlag, 1990: 476-501

[9] Schneier B. Description of a new variable - length key, 64 - bit block cipher (Blowfish) [C]// Fast Software Encryption-Porceedings of the Cambridge Security Workshop. Lectures Notes in Computer Science 809. Cambridge, United Kingdom, Springer-Verlag, 1994: 191-204

[10] Biham E, Biryukov A, Shamir A. Miss-in-the-middle attacks on IDEA, Khufu and Khafre [C]// 6th Fast Software Encryption Workshop, LNCS. vol. 1636, Springer-Verlag, 1999: 124-138

[11] Vaudenay S. On the weak keys of Blowfish [C]// Third International Workshop Proceedings. Springer-Verlag, 1996: 27-32

[12] Schneier B. Applied Cryptography [M]. 2 edition. John Wiley & Sons, Inc, Toronto, Canada, 1996

(上接第65页)

[13] Anceaume E, Gradinariu M, Ravoaja A. Incentive for P2P Fair Resource Sharing// Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P). 2005: 253-260

[14] Gong Haigang, Liu Ming, Mao Yingchi, et al. Research Advances in Key Technology of P2P-based Media Streaming. Computer Research and Development, 2005, 42(12): 2033-2040

[15] Tao Shaohua, Liu Yuhua, Xu Kaihua, et al. The Strategies against Vulnerability of Hubs in Complex Networks. Computer Engineering and Applications, 2007, 43(2): 151-153

[16] Policies for IPv4 address space management in the Asia Pacific region. http://www.apnic.net/trans/cns/ add-manage-policy.pdf

[17] Albert R, Barabási A L. Emergence of scaling in random networks. Science, 1999, 286: 509-512