面向分组密码算法的程序设计语言研究

李凤华[1],阎军智[2],谢绒娜[1],马建峰[2],欧海文[1]
(1.北京电子科技学院研究生处,北京 100070;
2.西安电子科技大学计算机网络与信息安全教育部重点实验室,陕西西安 710071)

摘 要: 本文提出了一种接近数学描述的面向分组密码算法的程序设计语言(Programming Language for the Block Cipher Algorithm,PLBCA).PLBCA 能够以形式化方式方便地描述分组密码算法的结构.本文介绍了 PLBCA 的语法规范,以分组密码算法 DES 为例说明 PLBCA 应用方法,并借助 ANTLR 工具实现了 PLBCA 的解析器.利用 PLBCA,密码学专家可以方便快捷地对密码算法进行算法正确性和安全性分析,以检验算法的设计.PLBCA 有助于提高密码算法检验的效率,为密码算法的设计和自动检测分析提供了一种辅助工具.

关键词: 领域专用语言; 分组密码算法程序设计语言; DES 密码算法
中图分类号: TP302.1 **文献标识码:** A **文章编号:** 0372-2112 (2009) 12-2705-06

Research on the Programming Language for the Block Cipher Algorithm

LI Feng-hua[1], YAN Jun-zhi[2], XIE Rong-na[1], MA Jian-feng[2], OU Hai-wen[1]
(1. *Graduate School, Beijing Electronic Science and Technology Institute, Beijing* 100070, *China*;
2. *Key Laboratory of Computer Networks and Information Security (Ministry of Education), Xidian University, Xi'an, Shaanxi* 710071, *China*)

Abstract: The Programming Language for the Block Cipher Algorithm (PLBCA) is proposed, which is similar to the mathematical description and is adapt to describe the structure of block cipher algorithm in a formal way. The grammar of PLBCA is presented and the DES algorithm is described using PLBCA as an example. The PLBCA parser is implemented by the ANTLR tools. Using PLBCA, the cryptography experts could test and analyze the cryptographic algorithms conveniently and quickly. PLBCA may improve the testing efficiency of cryptographic algorithm, and it can be used as an assistant for the design and automatic analysis for cryptographic algorithms.

Key words: domain specific language; programming language for the block cipher algorithm; DES

1 引言

随着信息技术的不断发展,通过网络传输获取信息,已从军事、政治、外交等重要领域日益普及到人们日常生活的各个领域,确保信息的安全变得刻不容缓.密码是保护信息安全的最有效的手段,也是保护信息安全的关键技术.密码算法对整个信息系统的安全性起着至关重要的作用.在进行密码算法研究与设计期间,密码学专家更多关心密码算法的设计思想与原理,希望有一种简单直观的程序设计语言,既能描述密码算法结构,又便于验证设计思想,减少编程设计的工作量,并避免因编程差错而引起的检验结果偏差等问题.

Cryptol[1,2]是一种由 Galois 公司开发的密码算法语言,可以将密码算法生成 C、C++ 以及 VHDL 等代码,主要用于设计、实现和验证密码算法.Cryptol 最初是为美国国家安全局(NSA)开发的分组密码算法语言,在开发初期参考了高级加密标准的五个候选算法(MARS、Serpent、TwoFish、RC6 以及 Rijndael),旨在为密码学以及密码算法应用提供一种规范语言[3].Cryptol 具有可执行性,利用 Cryptol,密码算法设计者可以在设计过程中直接执行算法.Cryptol 与平台无关,利用 Cryptol 描述现有的密码算法,开发者无需关注算法细节即可将算法快速实现,使用 Cryptol 描述的密码算法具有可重用和可移植性[1,3].有关 Cryptol 的使用方法详见文献[4].μCryptol[5]是由 Cryptol 衍生出的一种用于对称密码体制的密码算法语言,专门为没有动态内存管理功能的硬件和嵌入式微处理器而设计[6],但并未获得广泛应用.由于 Cryptol 侧重于密码算法的具体实现与应用,不是采

收稿日期:2008-12-26;修回日期:2009-03-31
基金项目:国家 863 高技术研究发展计划(No.2007AA01Z472,2007AA01Z429,2007AA01Z482); 国家自然科学基金(No.60633020,60702059);教育部重点项目(No.209156);北京市自然科学基金(No.4082028,4102056);天津科技攻关计划项目(No.06YFGZGX17500);北京电子科技学院信息安全重点实验室基金(No.YZDJ0807)

用类似数学的语言对密码算法进行描述,因此描述不够简单直观,代码易读性较差,不能直观地体现出密码算法的设计思想以及运算细节.然而,在众多应用场合,尤其在密码算法研究与设计过程期间,设计者需要使用一种简单直观的描述语言,不断与相关设计人员和测评人员交流算法的设计思想与细节,Cryptol 并不能满足这种需求.因此,需要为密码算法设计者提供一种简单直观的密码算法语言,用于描述密码算法的设计思想和运算细节,并用于检验算法的正确性和安全性,从而为密码学专家提供一种密码算法设计和自动检测的辅助工具.

本文以方便密码学专家描述分组密码算法为目的,提出了一种接近数学描述的分组密码算法的程序设计语言 PLBCA,并借助 ANTLR 工具实现了分组密码算法解析器.PLBCA 以自然的类似专业语言的思维方式描述算法的设计思想,其代码是对分组密码算法的直观描述,可以直接反映出算法的设计细节.因此,利用 PLBCA,设计者可以方便地与其他设计人员交流算法的设计思想.由于 PLBCA 简单易用,密码学专家无需懂得通用程序设计语言,即可方便快捷地实现算法,从而可以对分组密码算法进行算法正确性和安全性分析,以检验算法的设计.在密码算法测评阶段,若采用传统方法,即由设计者提供算法的 C 或 C++ 等通用程序设计语言的源代码,测评人员需要花费大量的精力研究算法运算细节,并检验代码的正确性,采用 PLBCA,测评人员能够方便快捷地验证代码的正确性,提高测评效率.与 Cryptol 相比,PLBCA 更接近于密码算法的数学描述,其主要目标是直观地描述密码算法,并生成密码算法的运算结果,更加侧重于语言的易用性.PLBCA 有助于提高密码算法设计和检验的效率,为密码算法的设计和自动检测分析提供了一种辅助工具.

PLBCA 和 Cryptol 都是密码算法的领域专用语言(Domain Specific Language,DSL).DSL 是特定于某个领域需求的描述语言[7,8],它具有特殊目的性、专业性和特定任务性,提供足够多的语法规范支持该领域中通用的各种概念,使该领域的工作者可以方便快捷地构造领域内的应用程序[9,10].开发 DSL 要求开发人员必须是开发领域和该语言使用领域的专家,但很少有人同时具备这两者,因而开发过程比较困难,发展也比较缓慢.近年来,由于 Ruby on Rails[11] 的出现和应用,使得 DSL 有了很大的发展,并在许多领域得到应用.

2 分组密码算法程序设计语言

PLBCA 是一种面向分组密码算法的专业描述语言,设计的目标是使密码学专家能以最自然的类似专业语言的思维方式描述算法的设计思想,书写整个算法的过程.在密码算法程序设计语言的设计过程中,一方面需要研究现有密码算法的特点及专用符号,并将其抽象表达出来,另一方面需要研究密码算法的普遍规律,使设计的密码算法程序设计语言既能描述现有密码算法,又能描述新设计的密码算法,同时也让密码学专家描述算法思想时接近密码算法数学表示的特征[12~14].此外,程序设计语言的词法要尽量简明、易于表达,语法成分应宜于描述密码学专业的表达.

2.1 PLBCA 的范式描述

以下采用扩展的 BNF 范式定义 PLBCA 的语法规范:

⟨program⟩ ::= [explanation_list]⟨program_block⟩
⟨explanation_list⟩ ::= "/explain"⟨explanationseq⟩" \ explain"
⟨explanationseq⟩ ::= *⟨explanation⟩
⟨explanation⟩ ::= ⟨var⟩"("⟨expression⟩")";"/⟨modular_dec⟩";"
⟨modular_ dec⟩ ::= ⟨var⟩"("⟨param_list⟩")"
⟨param_list⟩ ::= ⟨var⟩*(","⟨var⟩)/empty
⟨program_block⟩ ::= "/main"⟨statement_list⟩" \ main"
⟨statement_list⟩ ::= *⟨statement⟩/empty
⟨statement⟩ ::= ⟨expression_stmt⟩/⟨modular_stmt⟩/⟨iteration_stmt⟩/⟨selection_stmt⟩
⟨expression_stmt⟩ ::= ⟨expression⟩";"/";"
⟨modular_stmt⟩ ::= ⟨modular_ dec⟩";"
⟨iteration_stmt⟩ ::= "/loop""("⟨expression⟩";""step = "⟨simple_expression⟩")""= "⟨expression⟩⟨statement⟩" \ loop"
⟨selection_stmt⟩ ::= "/if""("⟨expression⟩")""then"⟨statement⟩" \ if"/"/if"⟨expression⟩"then"⟨statement⟩"else"⟨statement⟩" \ if"
⟨expression⟩ ::= ⟨var⟩"= "⟨expression⟩/⟨simple_expression⟩
⟨simple_expression⟩ ::= ⟨var⟩⟨operator⟩⟨var⟩/⟨modular_dec⟩
⟨operator⟩ ::= +/-/*///⟨⟨/⟨⟨⟨/⟩⟩/⟩⟩⟩/&/|/! /^/= =/! =/⟨/⟨=/⟩/⟩=/||
⟨var⟩ ::= ⟨ALPHA⟩*(⟨ALPHA⟩/⟨DIGIT⟩/"_")
⟨ALPHA⟩ ::= %x41-5A/%x61-7A
⟨DIGIT⟩ ::= %x30-39

2.2 PLBCA 的语法规范

2.2.1 程序结构

PLBCA 的书写格式主要包含 3 部分:注释部分、说明部分、主体部分.

注释部分用于描述附加说明信息,类似通用程序设计语言的注释,对单行的注释使用"//",对多行的注释使用"/*…*/",注释部分没有任何实质语义.在任何允许空格出现的地方,都允许出现注释.

说明部分用于描述变量或常量的定义,以及变量赋初值等.说明部分以"/explain"开始,以" \ explain"结束,这部分内容相当于 C 语言中的变量声明.注释部分和说明部分是可选的.

主体部分以"/main"开始,以" \ main"结束,主要用来描述密钥变换、加解密主过程,也可包含变量或常量的定义.算法描述的编写可在任何文本编辑器中进行,文件内容为不带任何版式格式的纯文本格式,命名符

合 Windows 文件名格式即可.算法描述以行的形式表示,每行以分号“;”作为结束符.

2.2.2 关键字

PLBCA 是面向分组密码算法的程序设计语言,因而在实现过程中需要定义一些 PLBCA 描述密码算法所需要的关键字,如 explain、main、if、then、else、loop、step、bits、mod 等.关键字不区分字母的大小写.

2.2.3 数据类型

在 PLBCA 中涉及到许多关于比特串移位、按位与、比特串衔接与拆分等操作,却很少涉及字符变量,因而 PLBCA 只提供三种变量类型:

(1)比特串:一般用于表示明文、密文、密钥、大整数等与二进制比特串相关的数据.

(2)整型:直接采用十进制表示即可,如 $s = 12$.

(3)逻辑型:真值只有 True 和 False,用 1 表示 Ture,用 0 表示 False.

在使用过程中,变量不需要指定类型,但需要指定位数,如 var(bits = 65),表示变量 var 为 65 位;缺省时为 32 位,如 var 或 var().

PLBCA 变量名的命名规则为:

(1)变量名区分字母的大小写;

(2)变量名不超过 20 个字符;

(3)变量名必须以字母开头,变量名的组成可以是任意字母、数字、下划线,变量中不能含有空格及标点符号等;

(4)所有关键字和操作符都不能作为变量名.

PLBCA 数组的命名规则为:

(1)数组名命名规则与变量名要求相同;

(2)数组名维数不需事先声明,其“[”和“]”必须配套,之间可以是变量名或数值,最后一个“]”后不能再有数字、字母及下划线,即“]”作为数组名的结束符.

2.2.4 控制结构

由于 PLBCA 采用接近数学描述的表达方式,因而整个分组密码算法描述都是按顺序执行.PLBCA 含有循环控制和条件判断结构.循环控制语句以“/loop”开始,以“\ loop”结束,其结构如下所示:

```
/loop(s = 1;step = 1) = 16
…  //迭代内容
\ loop
```

其中 s 为变量,初始值为 1,step 表示步长,16 为循环次数,即每循环一次,s 数值增加 1,共循环 16 次.

条件判断结构以“/if”开始,以“\ if”结束,其结构如下所示:

```
/if (判断条件)
then 语句 1
else 语句 2
\ if
```

如果判断条件成立,执行语句 1,否则执行语句 2.

2.2.5 操作符与置换模块

PLBCA 为了书写的方便以及语言的整洁,为用户提供了一套常用的操作符,操作符包含普通的数学运算符,即“+,-,*,/”,由于没有浮点数,除法操作的结果取小数点前的整数,其他三种数学运算符分别为普通的加法、减法和乘法.除此之外,PLBCA 还包括常用的密码学操作,如位运算、逻辑运算、对二进制流的连接和拆分,有限域中的运算等操作,如表 1 所示.若有多个运算符,则优先计算括号中的运算符,然后采用从左至右的运算次序.

表 1　常用操作符

操作符	功能说明	示例	示例功能说明
$\langle\langle$	按二进制位左移	$b = a\langle\langle x$	将 a 按二进制左移 x 位,结果赋给 b
$\rangle\rangle$	按二进制位右移	$b = a\rangle\rangle x$	将 a 按二进制右移 x 位,结果赋给 b
$\langle\langle\langle$	按二进制位循环左移	$a\langle\langle\langle x$	将 a 按二进制循环左移 x 位,结果赋给 a
$\rangle\rangle\rangle$	按二进制位循环右移	$a\rangle\rangle\rangle x$	将 a 按二进制循环右移 x 位,结果赋给 a
&	与	$c = a \& b$	若 a,b 都是逻辑型数据,则为逻辑与操作;若有一者为整型数据,则为按位与操作,结果赋给 c
|	或	$c = a \mid b$	若 a,b 都是逻辑型数据,则为逻辑或操作;若有一者为整型数据,则为按位或操作,结果赋给 c
!	非	$b = ! \ a$	对 a 按位取反,结果赋给 b
^	异或	$c = a \hat{} b$	若 a,b 都是逻辑型数据,则为逻辑异或操作;若有一者为整型数据,则为按位异或操作,结果赋给 c
= =	判断操作符两边的表达式是否相等	$a = = b$	若 a 和 b 相等,则该表达式值为 1,否则为 0
! =		$a ! = b$	若 a 和 b 相等,则该表达式值为 0,否则为 1
$\langle$	判断操作符两边表达式的大小	$a\langle b$	若 a 小于 b,则该表达式值为 1,否则为 0
$\langle$ =		$a\langle = b$	若 a 小于或等于 b,则该表达式值为 1,否则为 0
$\rangle$		$a\rangle b$	若 a 大于 b,则该表达式值为 1,否则为 0
$\rangle$ =		$a\rangle = b$	若 a 大于或等于 b,则该表达式值为 1,否则为 0
||	衔接比特串	$b = a_1(A_1) \mid\mid a_2(A_2) \cdots \mid\mid a_n(A_n)$	等号在“||”左侧时,“||”表示衔接操作符.将 $a_1, a_2 \cdots a_n$ 衔接,结果赋给 b,其中 $A_1, A_2, \cdots, A_n$ 分别表示 $a_1, a_2, \cdots, a_n$ 的位数
	拆分比特串	$b_1(B_1) \mid\mid b_2(B_2) \cdots \mid\mid b_n(B_n) = a$	等号在“||”右侧时,“||”表示拆分操作符.将 a 依次按照 $B_1, B_2, \cdots, B_n$ 的位数进行拆分,结果分别赋给 $b_1, b_2, \cdots, b_n$
mod	求模运算	$c = a \bmod b$	将 a 模 b 的值赋给 c

除常用操作符以外,PLBCA 根据分组密码算法的普遍设计规律[15,16],提供了一些常用的置换模块.由于大量基于 Feistel 网络的分组密码算法的安全性都依赖于 S 盒[17,18],又鉴于 P 置换在分组密码中的广泛应用[19,20],PLBCA 提供了 S 变换、P 置换以及 AES 中的字节替代、列混淆等置换模块.本文定义并实现了一些常用的密码算法置换模块,如表 2 所示.在众多置换模块的原型中,需要定义输入和输出位数,一方面是为了使模块具有可扩展性,另一方面是为了检验代码正确性的需要.以 S 变换 substitute 为例,如表 2 所示,该模块对 M 进行 S 变换,其中 a、b 分别表示 S 变换的输入和输出位数,{}中数据为 S 盒.根据 a 和 b 的赋值,解析器可以判断代码中 S 盒以及需要进行变换的数据 M 的位数是否正确,从而增加代码的正确性检验功能.

表 2 密码置换模块

名称	示例	示例功能说明
permute	c = permute(a, b){3,2,…,15}(M)	P 置换:对 M 进行 P 置换,置换结果放在 c 中,a 表示 M 的位数,b 表示 c 的位数,{}中内容为置换表,其中的数值可以是常量或变量
substitute	c = substitute(a, b){5, 6, …, 15}(M)	S 变换:对 M 进行 S 变换,变换的结果放置在 c 中,{}中内容为 S 盒,其中的数值可以是常量或变量.a 表示 M 的位数,b 表示 c 的位数
bytesubstitute	c = bytesubstitute(a, b){35, a6, …,15}(M)	字节替换:对 M 进行字节替换,替换的结果放置在 c 中,{}中内容为替换表,其中的数值可以是常量或变量.a 为 M 的位数,b 为 c 的位数
mixcolumn	d = mixcolumn(a, b, c)	列混淆:对 a 进行列混淆,其中 a,b 都是字节矩阵,采用 16 进制表示.矩阵 a 和矩阵 b 按有限域内乘法相乘即得混淆结果,将其赋给 d,c 是有限域既约多项式的系数
fmul	d = fmul(a, b, c)	有限域中的乘法:a,b 为整型数据或比特串,c 为有限域中既约多项式的系数,将 a 和 b 按照有限域中的乘法相乘,结果赋给 d

3 基于 PLBCA 的 DES 密码算法描述

根据上述 PLBCA 的语法规范,本节以 DES 为例,给出 DES 的 PLBCA 代码.

3.1 说明部分

说明部分以"/explain"开始,以"\ explain"结束,主要用于定义存放密钥、明文和密文的变量,从文件中读取密钥和一个分组的明文等:

```
/explain  //说明部分
  key(bits = 64);//变量 key 为 64 位,key 用于存放 DES 算法的密钥
  plaintext(bits = 64);//变量 plaintext 为 64 位,用于存放 DES 算法的
                       一组明文
  ciphertext(bits = 64);//变量 ciphertext 为 64 位,用于存放 DES 算法的
                        一组密文
  input("key.txt",key);//从 key.txt 文件中读入 64 位,存入 key 变量
                       中,相当于 key 赋初值,即从密钥文件中取出
                       密钥
  input("plaintext.txt",plaintext);//从 plaintext.txt 文件中读入 64 位,存
                                   入 plaintext 变量中,相当于 plaintext 赋
                                   初值,即从明文文件中取出一组明文
\ explain
```

3.2 主体部分

主体部分以"/main"开始,以"\ main"结束,主要用来描述 DES 算法密钥变换、加解密主过程.在主体部分首先进行密钥变换生成子密钥,子密钥以 subkey[i], i =1,…,16 表示,subkey[1](48)表示第一个子密钥,48 表示密钥的位数.

```
/main
//密钥变换
X(56) = permute(64,56){57,49,…,4}(key);
C(28)||D(28) = X(56);
C(28)〈〈〈1;
D(28)〈〈〈1;
subkey[1](48) = permute(56,48){14,17,…,32}(C(28)||D(28));
C(28)〈〈〈1;
D(28)〈〈〈1;
……
//加密主过程
win(64) = permute(64,64){58,50,…,7}(plaintext);
L(32)||R(32) = win(64);
//16 轮循环变换
/loop(s = 1;step = 1) = 16
X(48) = permute(32,48){32,1,…,1}(R(32));
Y(48) = X(48)^subkey[s](48);
B1(6) ‖ B2(6)||B3(6) ‖ B4(6) ‖ B5(6) ‖ B6(6) ‖ B7(6) ‖ B8(6) = Y
(48);
S1(4) = substitute(6,4){14,0,…,13}(B1(6));
……
Fp(32) = permute(32,32){16,7,…,25}(S1(4) ‖ S2(4) ‖ S3(4) ‖ S4(4)
‖ S5(4) ‖ S6(4) ‖ S7(4) ‖ S8(4));
LL(32) = R(32);
R(32) = L(32)^Fp(32);
L(32) = LL(32);
\ loop  //循环结束
ciphertext = permute(64,64){40,8,…,25}(R(32) ‖ L(32));
output(ciphertext,"ciphertext.txt");//密文输出到 ciphertext.txt 中
\ main  //加密结束
```

4 PLBCA 的实现

ANTLR(ANother Tool for Language Recognition)前身是 PCCTS,是一个语言识别工具[21].它可以接受含有语法描述的语言描述符,通过语法描述来构造自定义语言的识别器、编译器和解释器.同时 ANTLR 支持动作和返

回值,还可以根据输入自动生成语法树并可视化地显示出来.而用户所要做的就是给语法树附上简单的操作符和动作来告诉 ANTLR 如何构造 AST(Abstract Syntax Tree,抽象语法树),并自动生成相应的 Java、C++、C# 或者 Python 等解释程序[22,23].

在研究 PLBCA 可实现性时,我们借助 ANTLR 实现了 PLBCA 解析器,其实现过程如图 1 所示.该过程主要包括以下 3 部分:(1)将描述 PLBCA 的词法和语法规则转化为 xx.g 文件;(2)借助 ANTLR 工具将 xx.g 文件转换为 xx.c 文件,进而生成 PLBCA 解析器;(3)将描述密码算法的 PLBCA 文件、存储明文或密文内容的文件、存储密钥的文件作为解析器的输入,之后解析器将算法执行结果(密文或明文)输出.

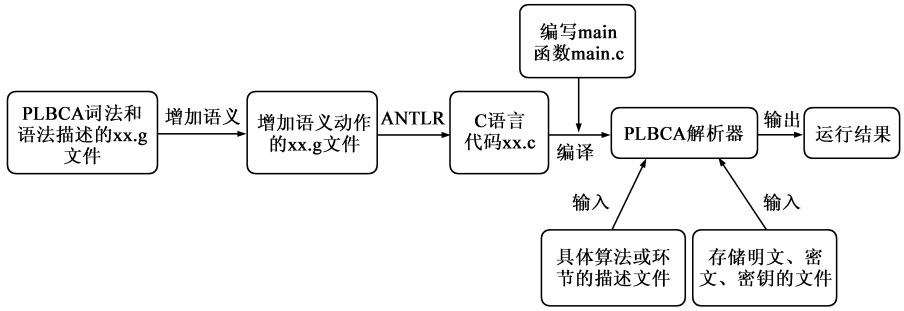

图1　PLBCA的实现流程

5　结束语

本文提出了一种面向分组密码算法的程序设计语言 PLBCA,并以 DES 密码算法为例对其进行了介绍,然后利用 ANTLR 实现了 PLBCA 的解析器.PLBCA 是一种接近数学描述的语言,可以直接反映出算法的设计细节.利用 PLBCA,密码学专家不再需要懂得通用程序设计语言,只需专注自己的密码算法设计,而不必再花费更多的时间和精力去关注算法的实现.PLBCA 为密码算法的安全性自动化检测分析提供了一种可行的方法,也为密码算法安全性等测评提供一种易用、高效的工具,具有重要的理论意义和现实意义.随着对 PLBCA 的深入研究,今后在此基础上将支持更多置换模块,以便能够描述更多的密码算法.

参考文献:

[1] J R Lewis, B Martin. Cryptol: high assurance, retargetable crypto to development and validation[A]. In Proceedings of the 2003 Military Communications Conference[C]. IEEE Press, 2003. 820 – 825.

[2] J R Lewis. Cryptol: specification, implementation and verification of high-grade cryptographic applications[A]. In Proceedings of the 2007 ACM workshop on formal methods in security engineering[C]. Virginia, USA: ACM, 2007. 41 – 41.

[3] Cryptol: the language for cryptography, Case Study[OL]. Available at http://www.galois.com/files/Cryptol/Cryptol_casestudy.pdf, Oct 2008.

[4] Cryptol: the language for cryptography, Programming Guide [OL]. Available at http://www.galois.com/files/Cryptol/Cryptol_Programming_Guide.pdf, Oct 2008.

[5] M B Shields. A language for symmetric-key cryptographic algorithms and its implementation[R]. Galois Connections Technical Report, 22 pages, Mar 2006.

[6] L Pike, M Shields, J Matthews. A verifying core for a cryptographic language compiler[A]. In Proceedings of the 6th international workshop on the ACL2 theorem prover and its applications[C]. Washington, USA: ACM, 2006. 1 – 10.

[7] M Marjan, H Jan, M S Anthony. When and how to develop Domain Specific Languages[J]. ACM Computing Surveys, 2005, 37: 316 – 344.

[8] K Toma, E M Pablo, A Pablo. A preliminary study on various implementation approaches of domain-specific language[J]. Information and Software Technology, 2008, 50(5): 390 – 405.

[9] A Deursen, P Klint. Domain-specific languages: an annotated bibliography[J]. ACM SIGPLAN Notices, 2000, 35(6): 26 – 36.

[10] D L Atkins, T Ball, G Bruns, et al. Mawl: A Domain-Specific Language for Form-Based Services[J]. IEEE Transactions on Software Engineering, 1999, 25(3): 334 – 346.

[11] D Thomas, D H Hansson. Agile Web Development with Rails, Second Edition[M]. Pragmatic Programmers, 2006.

[12] B Preneel, A Bosselaers. Recent developments in the design of conventional cryptographic algorithms[J]. LNCS, 1998, 1582: 106 – 131.

[13] J Daemen, R Govaerts, J Vandewalle. A new approach to block cipher design[A]. Fast Software Encryption[C]. Cambridge Security Workshop. LNCS, Vol. 809, 1994. 18 – 32.

[14] B Schneier. Cryptographic design vulnerabilities[J]. Computer, 1998, 31(9): 29 – 33.

[15] L R Knudsen. Block Ciphers-a Survey[J]. LNCS, 1998, 1582: 18 – 48.

[16] L R Knudsen. Block ciphers-analysis, design and applications [D]. PhD. Thesis, DAIMI PB 485, Aarhus University, Denmark, 1994.

[17] C Adams, S Tavares. The structured design of cryptographically good s-boxes[J]. Journal of Cryptology, 1990, 3(1): 27 – 41.

[18] 金晨辉,孙莹. AES 密码算法 S 盒的线性冗余研究[J].

电子学报,2004,32(4):639 – 641.
Jin Chen-hui, Sun Ying. Research on ther linear redundancy in the AES S box[J]. Acta Electronica Sinica, 2004, 32(4): 639 – 641. (in Chinese)

[19] J B Kam, G I Davida. Structured design of substitution-permutation encryption networks[J]. IEEE Transactions on Computers, 1979, 28(10): 747 – 753.

[20] L Brown, J Seberry. On the design of permutation P in DES type cryptosystems[A]. In Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology[C]. Houthalen, Belgium: Springer-Verlag, 1990. 696 – 705.

[21] T J Parr, R W Quong. ANTLR: A predicated-LL(k) parser generator[J]. Software - Practice and Experience, 1995, 25(7): 789 – 810.

[22] C Rod, H Paul. Create Domain Specific Languages with ANTLR[OL]. Available at http://www.devx.com/semantic/Article/35973? trk = DXRSS_JAVA, 2007.

[23] P Terence. An introduction to ANTLR[OL]. Available at http://www.cs.usfca.edu/parrt/course/652/lectures/ANTLR.html.

作者简介:

李凤华 男,1966 年 3 月出生于湖北省浠水县,西安电子科技大学博士生,北京电子科技学院教授,主要研究方向为网络安全与可信计算.
E-mail: lfh@besti.edu.cn

阎军智 男,1981 年 1 月出生于河南省郑州市,西安电子科技大学博士生,主要研究方向为密码学与网络安全.
E-mail: jzyan@mail.xidian.edu.cn

谢绒娜 女,1976 年 5 月出生于山西省永济市,北京电子科技学院助理研究员,主要研究方向为密码应用与信息安全.
E-mail: xierongna@besti.edu.cn

马建峰 男,1963 年 10 月出生于陕西省西安市,西安电子科技大学计算机学院院长、博士、教授、博士生导师,主要研究方向为密码学与网络安全.
E-mail: jfma@mail.xidian.edu.cn

欧海文 男,1963 年 6 月出生于黑龙江省海伦市,北京电子科技学院教授,主要研究方向为密码学与信息安全.
E-mail: ouhw@besti.edu.cn