面向代理重加密算法的程序设计语言研究

苏锰[1]，俞研[1]，吴槟[2,3]，付安民[1]

（1. 南京理工大学计算机科学与工程学院，江苏 南京 210094；
2. 中国科学院信息工程研究所信息安全国家重点实验室，北京 100093；
3. 中国科学院大学网络空间安全学院，北京 100049）

摘 要：通过结合领域专用语言（DSL），提出一种面向代理重加密的程序设计语言（PLPRE），PLPRE 支持代理重加密算法结构明确、类似数学语言的描述，算法设计人员通过 PLPRE 能够实现重加密算法快捷、方便的描述，并通过解析工具最终产生与之对应的计算机编程语言代码。首先介绍了 PLPRE 的语法规范，给出了关键字、程序逻辑的定义，并以 ACC-PRE 算法为基础，给出了 PLPRE 的描述示例，其次描述了基于 ANTLR 工具的 PLPRE 的解析与实现流程，最后通过与相关工作的对比，阐述了 PLPRE 的优势。PLPRE 的描述不需要关注数据结构、内存管理等问题，适用于不熟悉计算机编程的密码学家的描述算法，从而降低代理重加密算法研究中的设计与实现偏差。

关键词：代理重加密；领域专用语言；ANTLR；算法描述；密码算法实现

中图分类号：TP302

文献标识码：A

doi: 10.11959/j.issn.1000-436x.2018106

Research on the programming language for proxy re-encryption

SU Mang[1], YU Yan[1], WU Bin[2,3], FU Anmin[1]

1. School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
2. State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
3. School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: The programming language for proxy re-encryption(PLPRE) based on domain specific language(DSL) was proposed, which was similar to the matical description and was adopt to describe the proxy re-encryption directly. The algorithm designer could describe the algorithm conveniently and quickly by PLPRE, and obtain the programming code for computer. Firstly, the grammar of PLPRE was presented, including the definitions of the keyword and programming logic, then the ACC-PRE algorithm was described by PLPRE as an example. Secondly, the PLPRE parser was realized by ANTLR. Finally, the comparisons between the PLPRE and the related work were made. By using the PLPRE, user does not need to pay attentions to data structure or memory allocation. It is suitable for the cryptography experts who is not familiar with the programming. Thus, PLPRE will decrease the deviation between the design and implement for PRE.

Key words: proxy re-encryption, domain specific language, ANTLR, description of algorithm, implement of cryptography

---

收稿日期：2017-11-21；**修回日期**：2018-05-07

通信作者：吴槟，wubin@iie.ac.cn

基金项目：国家自然科学基金资助项目（No.61702266, No.U1536202, No.61572255）；江苏省自然科学基金资助项目（No.BK20150787, No.BK20141404）；北京市自然科学基金资助项目（No.4152048）；中国科学院信息工程研究所基础前沿基金资助项目（No.Y7Z0391102）；信息安全国家重点实验室重点布署基金资助项目（No.Y7D0061102）；中国科学院关键技术人才基金资助项目

Foundation Items: The National Natural Science Foundation of China (No.61702266, No.U1536202, No.61572255), The Natural Science Foundation of Jiangsu Province (No.BK20150787, No.BK20141404), The Natural Science Foundation of Beijing (No.4152048), IIE's Frontier Project (No.Y7Z0391102), SKLOIS Key Deployment Project (No.Y7D0061102), CAS Key Technology Talent Program

1 引言

云计算等复杂网络环境的发展，推动了数据共享和互联等相关技术的进步，越来越多的用户开始通过网络进行数据的传输和分发，其中，包含了大量隐私数据、机密数据的不可信的云存储环境增大了用户共享的难度，各类外包计算和云应用需求给人们带来了隐私泄露的风险[1]。为了确保该类数据的安全使用，保障用户和服务提供商双方的权益，数据加密和签名的相关技术被广泛应用。代理重加密是众多的密码技术之一，其依托于公钥密码加密的思想，采用公私钥对的形式进行秘密数据的发布和使用，基于其技术特点，密钥的产生可以由代理服务器进行处理，在实现共享的同时，减轻数据分享者的计算量[2]，这对于云这类用户众多、密钥计算量巨大的系统来说至关重要，因此，受到众多云计算安全研究人员的青睐。

代理重加密技术的研究涉及密码算法的设计和实现这 2 个方面，密码算法的研究和设计者多为密码学家或数学领域工作者，其描述的算法从数学理论方面进行分析和证明，在理论层面对其进行安全性分析，依托线性对、有限域等数学理论。算法一般采用形式化的方式进行描述，安全性证明则通过随机语言、标准等模型，基于“挑战—应答”的机制。算法在实现与具体应用时，需要针对具体系统进行程序代码编写，该方面的研究人员多为计算机专业，依托于程序设计理论，对于算法的描述通常采用流程图、伪代码等形式进行，最终编写具体程序语言代码。这个过程中除了关注数学理论外，更加注重数据定义、程序逻辑结构等方面，例如，数学语言使用无符号字符数组定义明文数据，而不是使用假设变量进行描述。

针对密码算法的实现问题，主流研究思想是通过开发开源密码算法代码库，为广大算法研究人员提供支持。目前，对于对称密码、非对称密码等传统密码算法均有对应的开源代码库，例如，crypto++[3]提供了 C++版本的多种传统密码算法的实现接口，包含 AES、RC6、MARS、Twofish、RSA、ECC 等，同时兼顾密钥分发、散列函数等多个方面。结合 SSL 协议的特点，OPENSSL 开源代码库为用户提供了对称加解密、数字签名等相关传统密码的应用接口。上述代码库提供了针对传统密码学算法的实现，对于目前的属性基加密、双线性对理论并未提供支持。为此，文献[4]设计并实现了针对双线性对的 C 语言代码库 PBC Library，提供了双线性对相关的定义、计算等数学运算的计算机代码。PBC 为用户提供了基本的群、有限域等运算，同时实现了线性对相关的运算，为广大密码研究人员的算法实现提供了重要程序基础[5]。随着各类计算机编程语言的发展，PBC 不再局限于 C 语言，出现了针对 Java 的 JPBC、针对 Python 的版本，将 PBC 的 C 语言接口进行了进一步封装，使用起来更加便利，面向的操作系统也涉及 Windows、Linux 等。更进一步，针对代理重加密，文献[6]提出了开源代码库 PRL，通过 C++或 C 语言进行代理重加密的基本核心代码，并且在不断扩充算法。PBC、PRL 等的出现在很大程度上方便了研究人员的算法实现与测试，但是 PBC 具体程序的编写过程中仍然需要熟悉具体程序语言、进行数组定义、明确内存空间分配原则等，这些对于密码算法设计人员仍然是一项不小的挑战。

上述研究成果与开源库为密码算法实现提供了重要的编程工具，但是，这些开源工具一般基于指定的程序设计语言，要求使用者能够熟练地进行计算机语言的使用，给密码算法的设计者带来使用上的困难，进而导致密码算法的性能和安全性测试分析脱离具体系统环境，停留在数学理论层面；而计算机编程人员虽然熟知程序设计，但是对密码算法设计中所包含的数学理论和描述较为陌生，在具体实现的过程中容易出现应为理解的偏差导致的实现与设计脱节的问题。上述问题是目前包含代理重加密算法等一系列密码算法设计与测试面临的主要困境。

领域专用语言（DSL, domain specific language）有别于通用程序设计语言，其针对具体的领域和问题，例如，HTML、SQL、YACC 等，这类语言在使用过程中，不需要使用者关注过多的计算机实现细节，更加接近人类的自然语言或数学描述语言，为此 DSL 被引入密码算法的描述和实现中[7]。Cryptol 就是其中的代表之一，用于指定加密算法的特定域语言。其设计之初是用来表达各类加密算法，以便可以有效地应用到硬件电路上，后来不断地被用于软件设备上，可行度较高。Cryptol 与平台无关，侧重算法的具体实现，其描述规则类似于算法的数学描述[8]。Cryptol 在密码算法描述

和实现方面被广泛应用于安全性和正确性的证明和测试[9-10]，但是该语言在直观性上仍然有待提高。文献[11]针对对称密码算法的特点，提出一种面向分组密码算法的程序设计语言 PLBCA，并针对 DES 算法进行了描述，该语言定义了变量、控制结构、置换函数等元素，对于对称密码算法能够进行类数学语言的描述，并基于 ANTLR（another tool for language recognition）进行了实现，通过 ANTRL 的解释，最终对对称密码算法进行实现。PLBCA 的思想即为密码学家提供密码算法正确性和安全性分析的具体实现途径，描述简明直观。上述 DSL 语言在密码算法实现方面的研究为代理重加密的算法实现提供了重要的理论支撑，若将 DSL 语言引入代理重加密的算法实现中，就为算法设计人员提供了一个类似数学语言的编程描述工具，搭建了一座算法设计与编程实现的桥梁。

本文针对上述代理重加密研究过程中面临的问题，提出一种针对代理重加密算法的程序设计语言（PLPRE, programming language for proxy re-encryption），通过近乎数学语言的方式对代理重加密算法进行描述，并借助 ANTLR 工具[12]对描述程序进行解析，产生对应的计算机语言代码，既能够适用于密码学家描述，又能够与计算机编程语言无缝对接。这有利于算法研究人员的交流和算法实际性能的测试与分析，减轻算法研究人员的计算机编程难度，同时避免计算机编程人员由于数学理论陌生而导致的实现偏差，将推动代理重加密理论在各类系统的广泛应用。

2　PLPRE 相关介绍

PLPRE 语言设计的初衷在于为密码研究人员提供一种面向重加密算法的专业程序设计语言，使密码学专家能够以近乎自然语言或数学语言的方式和思维进行算法的描述，以表述算法思想和过程。在 PLPRE 语言的研究过程中，一方面需要分析各类代理重加密算法的基本步骤，提取共性特点，并进行抽象的表达；另一方面，需要定义针对双线性对等理论和重加密算法的专用符号和运算。另外，PLPRE 的语法结构要尽量简单、明确、易于表达，同时要兼容 PBC 等开源代码库，实现代理重加密算法设计与实现的无缝对接。

2.1　范式定义

PLPRE 采用 BNF 范式的形式进行基本的语法规范描述，具体范式如下。

<program>::=<definition_list><program_block>
<definition_list>::=“/def”<definition_seq>“\def”
<definition_seq>::={<definition>}
<definition>::=<var>“(”<expression>“)”“；”
|<modular_dec>“；”
<modular_dec>::=<var>“(”<para_list>“)”
<para_list>::=<var>*(“,”<var>)|empty
<program_block>::=“/start”<function_list>“\end”
<function_list>::={<function>}
<function>::=“/method”<function_name>
“(”<para_list>“)”“<function_block>”“\method”
<function_name>::=“Setup”|“KeyGen”|“Enc”
|“ReKeyGen”|“ReEnc”|“Dec”|“main”
<function_block>::={<statement>}|empty
<statement>::=<expressions>|<modulars>|
<loops>|<selections>|<sub_fun>
<expressions>::=<expression>“；”
<modulars>::=<modular_dec>“；”
<loops>::=“/loop”“(”<expression>“；”“step=”
<simple_expression>“；”“until=”<expression>“)”
<statement>“\loop”<selections>::=“/if ”“(”<expression>“)”“then” <statement>“\if ”|“/if ”“(”<expression>“)”“then” <statement>“else”<statement>“\if ”
<sub_fun>::=<sfun_name>“(”<para_list>“)”
<sfun_name>::=“Prime”|“Group”|“Generator”
|“e”|“random”
<expression>::=<var>“=” <expression>|
<simple_expression>
<simple_expression>::=<var><op><var>|
<modular_dec>
<op>::=^|@|“xor”|==|!=
<var>::=<ALPHA>*(<ALPHA>|<DIGIT>|“_”)
<ALPHA>::=%x41-5A|%x61-7A
<DIGIT>::=%x30-39s

2.2　PLPRE 语法

2.2.1　程序结构

PLPRE 的程序主要包含定义、主体、注释 3 个部分。

定义部分用于对算法描述过程中涉及的变量或常量进行定义、对变量进行初始赋值等。定义以“/def”表示开始定义，以“\def”表示结束定义，该部分等同于其他程序语言中的声明、变

量定义的部分，对于代理重加密算法描述，定义的内容包含群、域、生成元、大素数等数学相关变量定义以及明文、密文等内容的定义，是必选部分。

主体部分以“/start”关键字标识主体描述开始，以“\end”关键字标识主体描述结束，主要用于代理重加密的算法描述，包含初始化、加密、重加密、密钥生成、重加密密钥生成、解密等主要函数过程，算法的描述可以使用普通的文本编辑进行，采用纯文本格式，算法描述以“行”为单位，以“;”标识“行”的结束。

注释部分是对定义、主体部分内容的附件说明，与现有程序设计语言的注释功能类似，单行以“//”标识开始，内容不能跨行；多行则以“/*”标识开始，“*/”标识截止，可以跨行。

2.2.2 关键字

PLPRE 需要关键字实现具体描述，关键字包括描述类、控制类和函数类，其中，描述类包含 def、method、sfun、bits、start、end 等，分别用于表示变量或常量定义、函数描述起始、子函数调用、数据位数定义、描述主体开始、描述主体结束；函数类包含 Setup、KeyGen、Enc、ReKeyGen、ReEnc、Dec、main 等关键字分别表示初始化、密钥生成、加密、重加密、重加密密钥生成、解密函数和主函数的定义；控制类包含 if、then、else、loop、step、until，分别针对选择和循环类的操作。

2.2.3 变量及常量类型

变量定义以关键字“/def”为起始，以“\def”为终止。代理重加密的算法描述中涉及循环群、双线性对、生成元、大素数等变量类型的定义，涉及的运算包含散列、按位逻辑运算等，多针对整数、比特串等进行描述。因此，PLPRE 所提供的变量类型如下。

整型：采用十进制表示。

逻辑型：仅包含 true 和 false 这 2 种类型，使用 1 和 0 与之对应。

比特串：用于明文、密文、密钥、大素数等相关数据的定义，使用二进制比特串表示。

枚举型：用于循环群等相关类型的表示和定义。

在使用过程中，变量的定义仅说明位数，类型的定义则通过具体描述中子函数和运算规则进行表示。例如，定义循环群 G_1 的语句为“/def G_1 \def”，循环群的表示则在具体函数描述中通过语句“G_1=*Group*(q)”进行说明，其中，q 为群 G_1 的阶，*Group* 为系统子函数，子函数说明见 2.2.5 节。

PLPRE 变量名的命名规则如下。

变量名不得与关键字、运算符重复。

变量名需要区分大小写，即变量 g 与 G 为不同的变量。

变量名不超过 20 个字符，超出则忽略 20 个字符之外的内容。

变量名必须以字母开头，可以包含字母、数字、下划线，其他字符则不能包含在变量名中。

对于数组类数据的定义，则以“数组名[数据长度]”的形式进行定义，数组名的定义需要符合变量名的定义约束，数据长度采用整型定义。

2.2.4 函数

函数的描述规则是 PLPRE 中的重要组成部分，依托的数据理论、面向的具体环境的不同可能导致代理重加密算法描述各异，但是通常的算法描述中包含初始化、密钥对生成、加密、解密、重加密密钥生成、重加密等函数步骤，因此针对不同的代理重加密算法中可以抽象出共性的函数步骤进行描述语言的定义，PLPRE 的函数描述均以“/method”关键字为起始，形式为“/method”函数名（参数表），以“\method”表示函数描述结束；参数表中参数数量为 0~N，N 为自然数，参数数量由描述用户需求来确定。

具体函数的定义形式和描述说明如表 1 所示。

2.2.5 运算规则

PLPRE 的运算规则定义包含子函数描述和运算符描述。其中，子函数描述以\sfun 表示，子函数是指数学算法设计过程中相关的通用数学函数，由现有的通用程序源码库支撑。

子函数描述包括大素数生成子函数 *Prime*()、循环群生成子函数 *Group*()、整数循环群生成子函数 *GroupZ*()、生成元获取子函数 *Generator*()、双线性对生成子函数 *e*()、随机选取群中元素子函数 *random*()、散列函数构造函数 *Hash*()，运算符包含幂运算^、连接运算@、有限域乘*，具体说明如表 2 所示。

2.2.6 控制结构

PLPRE 中包含了条件选择和循环 2 类控制结构，条件判定以“/if”开始，结构如下所示。

/if（逻辑判定条件）

表 1　　PLPRE 函数定义说明

名称	描述	示例
Setup	初始化函数，实现代理重加密算法参数的初始化，如产生循环群、双线性对、大素数等	/method *Setup*(*k*){…}\method，*k* 为参数初始化参数，通常为大素数位数或循环群阶等，初始化函数的参数可为多个
KeyGen	密钥对生成函数，产生代理重加密参与双方的公私钥对	/method *KeyGen*(*g*,*Zq*){…}\method，*g*、*Zq* 为初始化过程中已经产生循环群生成元和 *q* 阶整数群，产生的公私钥对的定义则在定义部分进行描述
ReKeyGen	重加密密钥生成函数，产生代理重加密密钥	/method *ReKeyGen* (sk_2,sk_1,pk_1){…}\method，其中，sk_2、pk_1、sk_1 为用户 2 的私钥及用户 1 的公私钥，该函数产生用户 1 为用户 2 共享数据的重加密密钥 $rk_{1\to2}$
Enc	加密函数，用于代理重加密的第一次加密	/method *Enc*(pk_1,*m*){…}\method，参数 pk_1 为用户 1 的公钥，该函数将描述实现将明文 *m* 使用 pk_1 进行第一次加密，产生密文 C_1
ReEnc	重加密函数，对第一次的密文进行重加密	/method *ReEnc*(C_1, $rk_{1\to2}$){…}\method，参数 C_1 为第一次加密产生的密文，$rk_{1\to2}$ 为代理重加密密钥
Dec	解密函数，分为 Dec_1 和 Dec_2，分别用于第一次加密密文和重加密密文的解密	/method Dec_1(C_1,sk_1){ … }\method 和 /method Dec_2(C_2,sk_2){ … }\method，C_1、C_2 表示第一次加密密文和重加密密文，sk_1 和 sk_2 表示用户 1 和用户 2 的私钥
main	主函数，用于用户的测试运行	/method *main*(){…}\method

表 2　　PLPRE 子函数及运算符说明

名称	示例	功能描述
Prime	\sfun *Prime*(*k*)	产生长度为 *k* bit 的大素数
Group	\sfun *Group*(*q*)	产生 *q* 阶循环群
Group	\sfun *GroupZ*(*q*)	产生 *Zq*
Generator	\sfun *Generator*(*G*)	获取群 *G* 的生成元
e	\sfun *e*(G_1,G_T)	产生群 G_1 到群 G_T 的双线性对
random	\sfun *random*(*Zq*)	选取群 *Zq* 中的任意元素
Hash	\sfun *Hash*(*A*,*B*)	构造集合 *A* 到 *B* 的散列映射关系
@	*a*@*b*	实现 *a* 和 *b* 的连接运算
xor	*a* xor *b*	实现 *a* 和 *b* 的异或运算
^	*g* ^*x*	进行 *g* 的 *x* 次幂运算
*	*a***b*	群元素 *a* 与 *b* 进行域内乘运算

then 程序语句 1

else 程序语句 2

条件判定多用于加密、解密过程描述中的先决条件判定的描述。

循环以“/loop”和“\loop”分别表示开始和结束，“step =”表示起始变量变化步长，“until =”表示循环结束条件。具体描述示例如下。

/loop (*a*=0; step =1; until = 10)

…//程序语句，即循环内容

\loop

该示例中 *a* 为实变量，步长为 1，10 为截止条件，每执行一次程序语句，*a* 加 1，直至 *a*=10，进行 11 次循环。循环语句多用于明文数据输入、密文数据输出、密钥数据产生等描述。

3　PLPRE 的描述示例

基于上述 PLPRE 语法描述，现以文献[12]中描述的代理重加密算法为实例，给出该 PRE 算法的 PLPRE 描述程序。

3.1　定义部分

/def　/*参数定义*/

q (*bits*=128) ;

/*定义长度 128 bit 的变量 *q*，用于大素数生成，大素数具体位数有初始化函数参数 *k* 确定*/

g ;

G_1;//定义变量 *g* 用于循环群生成元，G_1 表示循环群

…//此处定义其他生成元、循环群以及其他中间变量，方法同上

plaintxt(*bits*=128) ;

ciphertxt(*bits*=128) ;

/*定义明文数据 *plaintxt*、密文数据 *ciphertxt*，长度均为 128 bit，重加密基于公钥密码体制，明文、密文不需要估计按照长度分组，此处定义 128 bit 旨在便于数据读取*/

Input (*plaintxt*, “plaintxt.txt”);

Output (*ciphertxt*, “ciphertxt.txt”);/*描述明文和密文的输入和输出文件*/

\def

3.2　主体部分

参考文献[13]的算法，本节选取 ACC-PRE 的初始化、加密、重加密部分进行描述。

1) 初始化函数 *Setup*

该函数输入参数 k，定义大素数长度，生成 k 长度的大素数 p，q 阶循环群，构造散列函数映射关系，具体算法及描述语言对应关系实例如表 3 所示。

2) 第一次加密函数 *Enc*

该函数输入明文及用户 i 的公钥 pk_i，产生第一加密的密文 C_1。算法及程序设计语言对应关系实例如表 4 所示。

3)重加密函数 *Re-Enc*

该函数输入第一次加密密文 C_1 与重加密密钥 rk，产生可由用户 j 解密的重加密密文 C_2。具体算法及描述语言对应关系实例如表 5 所示。

由于篇幅关系，密钥生成、重加密生成函数以及解密函数将不再赘述，具体描述方法与上述算法描述相同。

4　PLPRE 系统实现

ANTLR 是一种语言识别工具，前身为 PCCTS，该工具可以为包括 Java、C++、C#在内的语言提供了一个通过语法描述来自动构造自定义语言的识别器（recognizer）、编译器（parser）和解释器（translator）的框架，涉及词法分析器、语法分析器、语法分析树等内容。本文基于 ANTRL 工具，对

表 3　　ACC-PRE Setup 算法 PLPRE 描述

ACC-PRE 数学描述	PLPRE 描述
$Setup(k)\rightarrow param$，选取长度为 k 的素数 p，群 G_1、G_2 为 q 的乘法循环群，g 为 G_1 的生成元，散列函数组 H_1、H_2、H_3、H_4，其中，$H_1:\{0,1\}^*\rightarrow G_1$，$H_2:\{0,1\}^*\rightarrow Z_p^*$，$H_3:G_2\rightarrow\{0,1\}^l$，$H_4:\{0,1\}^*\rightarrow G_1$。公开参数 $param=\{p,G_1,G_2,g,H_i(i=1,\cdots,4)\}$。定义双线性映射 $e:G_1\times G_1\rightarrow G_2$	/method *Setup*(k) q=\sfun *Prime*(k);　/*k 长大素数*/ G_1=\sfun *Group*(q); /*q 阶循环群*/ G_T=\sfun *Group*(q); *Zq*=\sfun *GroupZ*(q); g=\sfun *Generator*(G_1); /loop (s=1;*step*=1; *until*=2) {$H_1[s]$=\sfun *Hash*(*str_b*,G_1);} H_2=\sfun *Hash*(*str_b*,*Zq*); H_3=\sfun *Hash*(G_T,*str_l*); /*构造散列映射关系，*str_b* 表示随意比特串，*str_l* 表示 l 长比特串，在定义中通过 *str_l*(*bits*=l)，进行说明*/ \sfun e(G_1,G_T);/*构造双线性对*/ \method

表 4　　ACC-PRE 第一次加密算法 PLPRE 描述

ACC-PRE 数学描述	PLPRE 描述
$Enc(m,pk_i)\rightarrow C_i$，用户 i 使用自身公钥 pk_i 加密明文信息 m，选取 $k\in G_2$，计算 $r=H_2(m\|k)$，则 $C_i=(c_1,c_2,c_3,c_4,c_5)$。 $c_1=g^r$; $c_2=ke(pk_i,H_1(pk_i))^r$; $c_3=m\oplus H_3(k)$; $c_4=H_1(pk_i)$; $c_5=H_4(c_1\|c_2\|c_3\|c_4)^r$	/method *Enc*(*Zq*,g,g_1,pk_1,m,C_1) k=\sfun *random*(GT); /*在 q 阶整数循环群 GT 上随机选取 k*/ r= $H_2(m\|k)$ $C_1[1]$=g^r；//密文第一部分，计算 g^r mod q $C_1[2]$=k*$e(pk_1,H_1[1](pk_1))$^r//密文第二部分，m 与 $e(pk,H_1[1](pk))r$ mod q 连接 $C_1[3]$=m@$H_3(k)$；//密文第三部分 $C_1[4]$=$H_1[1](pk_1)$; $C_1[5]$=$H_1[2](C_1[1]\|\|C_1[2]\|\|C_1[3]\|\|C_1[4])$^$r$; \method

PLPRE 语言及其解析程序进行了实现和测试，具体实现流程如图 1 所示。

表 5　**ACC-PRE 重加密算法 PLPRE 描述**

ACC-PRE 数学描述	PLPRE 描述
$Re\text{-}Enc(C_i, rk_{i\to j}^{condition}) \to C_j$，加密代理对密文 C_i 进行重加密，生成可以被 sk_j 所解密的密文 $C_j = (c_1', c_2', c_3', c_4', c_5')$ 。若 $e(c_1, H_4(c_1 \| c_2 \| c_3 \| c_4)) = e(g, c_5)$，则进行以下计算： $c_1' = c_1$; $c_2' = c_2(pk_j^r g^{-r}, H_1(pk_i)^{-sk_i}) e(pk_j^r, H_1(pk_j \| condition) H_1(pk_i)^{-sk_i})$ $= ke(pk_j^r, H_1(pk_j \| condition))$; $c_3' = c_3$; $c_4' = H_1(pk_j)$; $c_5' = H_4(c_1', c_2', c_3', c_4')^r$	/method *Re-Enc*(*rk*,C_1,*para*) /**rk* 为重加密密钥，有 *reKeyGen* 函数产生，一个包含 pk_2 及相关参数的一个枚举类型*/ \if (*e*(C_1[1], H_4(C_1[1]‖C_1[2]‖C_1[3]‖C_1[4]))=*e*(*g*,C_1[5])) then {C_2[1]=C_1[1]; C_2[2]=*k***e*(pk_2^*r*,H_1[1](pk_2‖*para*)); /**para* 对应元算法描述中的 *condition*，不同的算法针对的环境不同，因此会出现用户定义的参与，PLPRE 中统一约束为比特串*/ C_2[3]=C_1[3]; C_2[4]=H_1[1](pk_2); C_2[5]=H_1[2](C_2[1]‖C_2[2]‖C_2[3]‖C_2[4]); \method

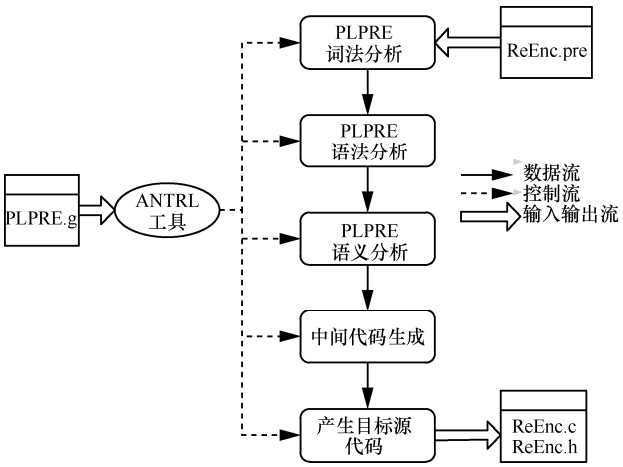

图 1　PLPRE 实现流程

用户依据自身需求，描述重加密算法，产生描述文本文件 ReEnc.pre，具体解析过程说明如下。

1) 对重加密描述文件 ReEnc.pre 进行词法分析。

步骤 1　预处理，去除无效的空格、制表符、换行这些字符。

步骤 2　识别专用运算符，如“^”“xor”“==”“!=”“@”，专用运算符识别为相应的 TOKEN 码。

步骤 3　识别关键字，为其指定相应的属性值，并将其对应为 TOKEN 码，为语法分析程序准备。

步骤 4　识别字符串、数字、字符常量，保存并将其对应为 TOKEN 码，为语法分析程序准备。

步骤 5　识别 main 函数描述结束符，并交给语法分析程序处理。

2) 接下来，对代理重加密描述文件进行语法分析，根据前面类数学描述语言的语法规则，把词法分析的结果分解为各个语法单元，并进行进行语法错误的检查与识别。

语法分析主要依托的是词法分析中产生的 TOKEN 码表，识别各类语法成分，分别包含关键字、变量、常量、函数、运算子函数和运算符，并进行关键字单词撰写错误、括号不匹配错误的识别。

3) 对代理重加密描述文件进行语义分析，并对描述语言具体语义含义进行识别和分析、静态语义检查，例如，指定变量是否定义、类型是否匹配、代理重加密步骤函数是否描述完整、齐全，为代码生成阶段搜集相关的语义信息。

经过语义分析将构造以下信息：全局变量、常量信息表；步骤函数信息表；步骤函数声明信息表；子函数调用信息表；产生中间代码。

4) 此时生成的中间代码已经类似于用户指定的计算机编程语言对应的代码，但是具体子函数的实施还未进行处理。

5) 依据用户的目标代码需求，选择对应计算机编程语言的代码底层库，生成目标代码。目前程序实现代码库依托于 PBC 库。在现有的 PBC 底层支持库中进行选择，调用不同的子函数实现程序及相关代码，并产生对应的头文件

函数生成列表。

6) 将产生的子函数代码与头文件信息进行整合，生成输出文件 ReEnc.c 和 ReEnc.h 文件，此处以 C 语言为例，若用户选择其他编程语言，则后缀名与文件格式会发生相应的变化。

上述 6 个步骤是依托于 ANTRL 工具产生的编译程序，ANTRL 工具则以 PLPRE.g 文件中描述的文法规则进行编译框架程序的生成，依据文献[12]，对第 3 节中定义的规则进行描述，针对变量描述的文法描述示例如下。

```
grammar PLPRE;
options {
    language = C;}
@header {
    #include <pbc.h>
    #include <gmp.h>}
expr: multExpr ((CON| EXP| XOR) multExpr)*;
CON: '@';
EXP: '^';
XOR:'xor';
multExpr : atom*
atom: INT
    | ID
    | '(' expr ')'!;
stmt: expr NEWLINE -> expr| ID ASSIGN expr NEWLINE -> ^(ASSIGN ID expr)| NEWLINE ->;
ASSIGN: '=';
ID: ('a'..'z'|'A'..'Z')+ ;
INT: '~'? '0'..'9'+ ;
NEWLINE: '\r'? '\n' ;
```

结合 3.1 节中的范式描述，进一步描述关键字、函数、子函数以及程序逻辑的文法描述，最终产生 PLPRE.g 文件。该文件通过 ANTRL 工具编译，产生 PLPRE 对应的词法、语法和语义分析器，用于算法源文件的编译。

5 PLPRE 特征分析

PLPRE 语言针对 PRE 的基本特征，提取功能步骤，给出了 PRE 算法描述的基本框架和语法结构，描述语言类似数学语言，对于密码算法研究人员和数学专家来说简单明了，本节将从知识背景、适用范围、跨平台等方面对 PBC、PRL、Cryptol、PLPRE 等进行对比，如表 6 所示。

1) 知识背景

PBC 与 PRL 基于 C++、C、Java 等语言，需要用户熟知计算机编程语言，明确数据的定义类型、长度，还需要用户在实现过程中进行数据结构、内存分配等方面的设计与分析；Cryptol 与 PLPRE 则仅仅关注数据的长度，不需要过度关注类型，更不用考虑如何分类内存和定义何种数据结构。对于非计算机从业的密码研究人员更为便捷。

2) 适用范围

Cryptol 主要针对传统密码算法，如 DES、AES，同时包含 MD5 等散列函数的描述方法，不适用于代理重加密算法的描述；PBC 则针对大量密码算法中至关重要的双线性对进行了 C 语言的实现，提供循环群产生、生成元提取、有限域运算等编程接口；PRL 更进一步针对 PRE 给出 C++程序接口，但是 PBC 与 PRL 需要在调用接口前，用户自行定义数据结构，设计程序逻辑；PLPRE 是针对 PRE 的专用 DSL，更加适合代理重加密算法的设计人员使用。

3) 跨平台

PRL 基于 PBC 进行了部分线性对重加密算法的 C++接口实现，需要运行平台具备 C++代码编译的能力；PBC 兼顾了 C++、Java 等多种高级语言，提供了具体的线性对相关的算法实现接口，但是仍然需要考虑具体选择语言适用的

表 6 相关技术对比

技术	知识背景			适用范围			跨平台
	需要关注数据类型	需要关注数据结构	需要计算机编程基础	支持 pairing	支持 no-pairing	支持重加密算法描述	
PRL	Yes	Yes	Yes	Yes	No	Part	No
PBC	Yes	Yes	Yes	Yes	No	Part	Part
Cryptol	No	No	No	No	No	No	Yes
PLPRE	No	No	No	Yes	Yes	Yes	Yes

平台；Cryptol 和 PLPRE 则具备了其特有的语法描述体系，并产生出具体编程平台需要的算法代码。

6　结束语

代理重加密以其特有的技术特点，被广泛应用于云安全、隐私保护等领域。通常从事代理重加密算法研究的多为熟知数学理论的非计算机专业人员，使算法设计仅通过数学分析进行安全性的证明，性能的分析则依托于理论计算；算法在具体应用时，系统编程人员由于不熟悉数学理论，往往出现理解的偏差，从而产生实现过程中的漏洞。这一现状导致了代理重加密算法的设计与实际应用脱节。本文针对这一问题，提出一种面向代理重加密算法的程序设计语言 PLPRE，该语言能够通过类似数学语言的方式对代理重加密算法进行描述。描述过程中不需要关注代码设计细节，适用于非计算机专业的密码算法设计者，从而避免代理重加密算法的设计与实现的偏差。这有利于算法研究人员的理论交流，减轻算法研究人员的计算机编程难度，进而推动代理重加密技术在各类系统的广泛应用。

参考文献：

[1] 张玉清, 王晓菲, 刘雪峰, 等. 云计算环境安全综述[J]. 软件学报, 2016, 27(6): 1328-1348.
ZHANG Y Q, WANG X F, LIU X F, et al. Survey on cloud computing security[J]. Journal of Software, 2016, 27(6): 1328-1348.

[2] 苏铓, 史国振, 谢绒娜, 等. 面向移动云计算的多要素代理重加密方案[J]. 通信学报, 2015, 36(11): 73-79.
SU M, SHI G Z, XIE R N, et al, Multi-element based on proxy re-encryption scheme for mobile cloud computing[J]. Journal on Communications, 2015, 36(11): 73-79.

[3] SIDOROV E, Breaking the Rabin-Williams digital signature system implementation in the Crypto++ library[J]. Lecture Notes in Computer Science, 2015, 2664(4):956.

[4] BENGER N, SCOTT M. Constructing tower extensions for the implementation of pairing-based cryptography[C]//Springer Berlin Heidelberg. 2012:180-195.

[5] LYNN B. On the implementation of pairing-based cryptosystems [J]. Dissertation Abstracts International, 2007,68:3903-3910.

[6] ATENIESE G, FU K, GREEN M, et al. Improved proxy re-encryption schemes with applications to secure distributed storage[J]. ACM Transactions on Information & System Security, 2006, 9(1):1-30.

[7] LEE P, MARK S, MATTHEWS J. A verifying core for a cryptographic language compiler[C]//International Workshop on the ACL2 Theorem Prover and ITS Applications. 2006: 1-10.

[8] LEVENT E, MATTHEWS J. High assurance programming in Cryptol[C]//The 5th Annual Workshop on Cyber Security and Information Intelligence Research. 2009: 60-61.

[9] AHMAD J J, LI S, SADEGHI A R, et al. CTL: a platform independent crypto tools library based on dataflow programming paradigm[C]// International Conference on Financial Cryptography and Data Security. 2012: 299-313.

[10] TOMB A. Automated verification of real-world cryptographic implementations[J]. IEEE Security & Privacy, 2016, 14(6): 26-33.

[11] 李风华, 阎军智, 谢绒娜, 等. 面向分组密码算法的程序设计语言研究[J]. 电子学报, 2009, 37(12): 2705-2710.
LI F H, YAN J Z, XIE R N, et al. Research on the programming language for the block cipher algorithm[J]. Acta Electronica Sinica, 2009, 37(12):2705-2710.

[12] BOVET J, PARR T. ANTLRWorks: an ANTLR grammar development environment[J]. Software Practice & Experience, 2008, 38(12): 1305-1332.

[13] SU M, LI F H, SHI G Z, et al. A user-centric data secure creation scheme in cloud computing[J]. Chinese Journal of Electronics, 2016, 25(CJE-4): 753-760.

[作者简介]

苏铓（1987-），女，内蒙古赤峰人，博士，南京理工大学讲师，主要研究方向为云计算安全、访问控制、代理重加密等。

俞研（1972- ），男，吉林长春人，博士，南京理工大学副教授，主要研究方向为无线网络、网络空间安全等。

吴槟（1980-），男，山东青岛人，博士，中国科学院信息工程研究所副研究员，主要研究方向为网络与信息系统安全、信息对抗理论与技术等。

付安民（1981-），男，湖北通城人，博士，南京理工大学副教授，主要研究方向为云安全、隐私保护等。