Computer Science and Application 计算机科学与应用, 2019, 9(1), 157-165
Published Online January 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.91019

Hans汉斯

Domain-Specific Language Research in the Field of Cryptography

Mingyu Fan*, Xinmin He

School of Computer Science and Engineering, University of Electronic Science and Technology of China (Network Space Security College), Chengdu Sichuan
Email: *ff@163.com, h20031795@vip.163.com

Received: Jan. 1st, 2019; accepted: Jan. 16th, 2019; published: Jan. 23rd, 2019

Abstract

It is difficult for beginners to write a simple loop structure with the existing general-purpose programming language. Domain-specific programming languages (DSL) are widely used to enable professionals to focus their efforts on application logic/business models. This paper analyzes the principles and methods of DSL, its maintenance issues, and introduces a cryptographic domain-specific language called PCL that was firstly proposed in China. The technical principle, basic functions, and application examples of cryptographic algorithms are given. Prior to PCL, there were no languages to describe the logical structure of cryptographic algorithms visually, including the Cryptol language.

Keywords

Information Security, DSL, PCL, PCL for Cryptography, Cryptol DSL

密码领域专用语言研究

范明钰*，何新民

电子科技大学计算机科学与工程学院(网络空间安全学院)，四川 成都
Email: *ff@163.com, h20031795@vip.163.com

收稿日期：2019年1月1日；录用日期：2019年1月16日；发布日期：2019年1月23日

摘　要

现有的通用编程语言，对于初学者来说，即使是编写简单循环结构的也有难度。为了使专业人员将工作重心集中在应用程序逻辑/业务模型上，领域专用编程语言(DSL)得到广泛应用。本文分析DSL的原理和

*通讯作者。

文章引用: 范明钰, 何新民. 密码领域专用语言研究[J]. 计算机科学与应用, 2019, 9(1): 157-165.
DOI: 10.12677/csa.2019.91019

方法，维护性问题；介绍一款国内首次提出的密码领域专用语言PCL，给出了其技术原理、基本功能，以及密码算法应用例子。在PCL之前，包括Cryptol语言在内，没有任何语言可以直观描述密码算法的逻辑结构。

关键词

信息安全，DSL，PCL，密码领域专用语言，Cryptol DSL

1. 前言

现有的通用编程语言，对于初学者来说，即使是编写简单循环结构的也有难度，因为有多种不同的方法可以完成同样的任务要求。这使得通用编程语言的使用者迷失于其语言功能，不能将工作重心集中在应用程序逻辑/业务模型上。

领域专用语言(Domain-Specific Languages，简称 DSL)是一种编程语言[1] [2] [3]，旨在为给定的问题域生成解决方案(与通用编程语言相对)。这样可使领域专家按照其熟悉的术语设计方案，在一个对其最有意义的抽象层次上来完成设计。大多数 DSL 具有基于解释器的“开发环境”以直接执行用于实验目的的程序。此外，由于 DSL 往往比通用编程语言小，因此通常更容易添加新的目标后端，生成直接在目标体系结构上运行的可执行文件。DSL 还可以用于描述异构系统，并为这些不同目标生成不同的代码。DSL 为良好的工具支持开辟了道路：模拟设计探索，自动测试和自动生成测试工具，为多个目标生成高度专业化的代码，以及生成正确、安全属性的正式证据。

与从头开始实现任务相比，使用 DSL 工具是否会提高语言实现的可维护性？文献[4]通过分析使用不同语言(Java, JavaScript, C＃)和 DSL 工具(ANTLR [5], OMeta [6], Microsoft “M”)的六个相同 DSL 实现的可维护性方面的实证结果。评估表明，使用 DSL 工具构建语言实现的可维护性确实更高。

本文分析领域编程语言的基本要求，及其在密码学领域中的应用，并研发了一款国内首次提出的密码领域专用语言 PCL (Point Cryptographic Language)。

2. DSL

在图灵提出其计算模型几十年之后，我们仍然缺乏合适的方法来解决计算机程序人机交互的复杂性。在科学和工程涉及的解决方法中，可以分为通用方法和特定方法。通用方法为某个领域的众多问题提供一般解决方案，但这种解决方案可能不是最理想的。特定方法则为较少的问题提供更好的解决方案。

过去的编程语言(如 Cobol [7]，Fortran [8]，Lisp [9]等)最初都是作为专用语言出现的，用于解决某个领域的问题(分别是业务处理，数值计算和符号处理)。渐渐地，就演变成通用语言，并且经过一再重复使用后，就需要更专业的语言支持方法来解决应用程序域中定义良好的问题。归纳起来，随着时间的推移，采用以下方法来解决专业问题：

1、子程序库包含子程序，这些子程序在明确定义的域中执行相关任务，例如，微分方程，图形，用户界面和数据库。子程序库是打包可重用域知识的经典方法。

2、面向对象的框架和组件框架，延续采用子程序库的概念。经典库具有扁平结构，应用程序调用库。

在面向对象的框架中，通常情况是框架处于控制之下，并调用由特定于应用程序的代码提供的方法。

DSL 是在领域专家之间进行沟通、构建模型和提高开发效率的专用语言工具，既要保证既定领域中常规性内容可以轻松地表达，又要为原本费神的复杂结构提供简洁的语法支持。

我们定义，域特定语言(DSL)是一种编程语言或可执行规范语言，其通过适当的符号和抽象提供集中于并且通常限于特定问题域的表达能力。根据这个定义，DSL 的关键特性是它们集中的表现力。

尽管多年来已经设计出了多种特定领域的编程语言，但是对于特定领域编程语言的系统性研究也才起步[1] [2] [3] [4]。

特定于域的语言(DSL)，是一种小型的，通常是声明性的编程语言，具备提供特定问题域的表达能力。在许多情况下，DSL 程序被转换为对公共子程序库的调用，并且 DSL 可以被视为隐藏程序库细节的手段。

DSL 通常很小，只提供一套有限的符号和抽象。在文献中，它们也被称为微语言和小语言。然而，有时它们也包含整个通用语言(GPL)作为子语言，因此除了 GPL 的表达能力之外还提供特定于域的表达能力。当 DSL 作为嵌入式语言实现时会出现这种情况。诸如 Cobol 或 Fortran 之类的语言，通常不被视为 DSL，而分别视为针对商业和科学领域定制的编程语言，因为它们不小并且表达能力不限于这些领域。

2.1. DSL 的开发

如前所述，领域专用语言通常是声明性的，因此，可以视为规范的编程语言。根据生成结果，有至少三种类型的 DSL 开发方式：

1) 生成应用程序的 DSL，利用 DSL 编译器，生成应用程序。这种 DSL 编译器在文献[4]中又被称为应用程序生成器，而这种 DSL 被称为应用程序专用语言。

2) 生成库或组件的 DSL，例如 YACC [10]或 SDL [11]，并非旨在编程(指定)完整的应用程序。

3) 生成文档或图片的 DSL。

本文仅涉及第一种 DSL。

用于构建业务数据处理系统的 DSL 的常用术语是第四代编程语言(4GL)。与特定于域的编程相关的是最终用户编程，当最终用户使用宏或脚本语言执行简单的编程任务时会发生这种情况。一个典型的例子是使用 Excel [12]宏语言的电子表格编程。

有多种方法开发 DSL。内部(嵌入式) DSL 开发方法，采用重用通用主机语言(例如，Ruby [13]，Scala [14])的语法和解析器。外部 DSL 开发方法，使用 DSL 工具，这些工具包括解析器生成器(例如，ANTLR [5]，Yacc [10]等等)，转换系统(例如，ASF + SDF [15]，Stratego [16]，TXL [17])或属性语法系统(例如，JastAdd [18]等)。这些工具的目标是降低构建 DSL 的成本。

就像软件必须发展以保持可行一样，DSL 也受到维护活动的影响[4]。与主流软件应用程序相比，DSL 实现特别复杂且繁琐。因此，DSL 实施的可维护性似乎比平常更重要。我们还观察到，即使使用 DSL 工具，维护编程语言实现也需要在实践中付出相当大的努力。因此，主要目标是研究使用 DSL 工具对最终实现的可维护性的影响的重要性。

2.2. DSL 的可维护性

软件维护由 IEEE 定义如下[19]：在交付之后修改软件系统或组件以纠正故障，改进性能或其他属性或适应变化的环境的过程。

维护可以进一步分为纠正，修复缺陷；自适应，响应变化的环境；完善，例如，改善性能或其他属性；和预防性维护，检测和纠正未发现的缺陷[19]。

文献[4]对同一种 DSL 的六种实现进行了初步的实证研究。三个实现分别从头开始，分别用 Java，

JavaScript 和 C＃开发。这些实现称为“vanilla”实现。其他三种实现由 DSL 工具支持：ANTLR(Java) [5]，OMeta (JavaScript) [6]和 Microsoft SQL Server 建模平台的“M” (C＃)。

这些实现代表 DSL 的常见实现。vanilla 实现使用通用设计模式进行语言实现。各个 DSL 工具实现已经分别由 ANTLR 和 OMeta 的作者以及 Microsoft Nederland 的“M”专家进行了审查。

与不使用 DSL 工具相比，使用 DSL 工具是否有利于编程语言实现的可维护性？

如果假设，由于维护是由理解 DSL 工具的且经验丰富的工程师完成的，因此这个问题的答案是肯定的[4]。实验评估旨在使该假设无效。考虑使用 Yacc 实现 C++解析器所必需的动作代码的广泛使用：同一解析器的 vanilla 实现可以更容易维护。

1) 评估六个以上与开源软件相同的非平凡 DSL 的实现。实现都包括解析器，检查器和评估器。

2) 提出其中六种实施的定量数据(指标)和定性分析。

3) 结果在很大程度上证实了使用 DSL 工具提高可维护性的假设。

研究结果明确地没有给出任何关于是否应该使用某种工具的建议，因为 DSL 实现的质量有很多方面；可维护性只是其中之一。我们将性能、灵活性(例如，错误处理)和可用性(例如，调试，测试等)视为未来工作的方向。

3. 密码领域的 DSL

目前，加密应用程序大都采用传统的通用编程语言编写，如 C/C++。这些语言没有支持加密算法测试和诊断的内置功能。

密码领域专用语言与其他计算机通用语言相比，其显著特点与密码算法的基本构成相关，包括几方面：其一是数据和运算，密码算法定义的数据如移存器、S 盒、任意数据规模的序列，以及定义其上的抽取、置换、截断、填充等非常专业的运算；其二是逻辑结构，密码算法本身非常强的逻辑结构如分层、对称、迭代、混淆等。此外，还有密码算法特有的精准描述和正确性、一致性验证。而与其他计算机编程语言不同的是，大多数计算机程序应用中的 BUG 可以在使用过程中发现并修正，而密码算法逻辑中如果存在 BUG，其后果严重且难以弥补。

当以通用语言实现加密算法时，由于缺乏从概念到代码的更直观的转换的构造，会面临许多问题。

在国内，描述密码算法的工具大多为 C 语言主宰。但是，C 语言并不适合密码算法的描述，例如，由于缺乏适合移位寄存器的数据类型，为了用 C 语言的现有类型拼接出一个移位寄存器，就有多种多样的拼接方法。又例如，加密算法需要大量处理对于数据的捕获操作。以位操作为例，程序员需要管理是数据的位而不是字节。虽然在大多数编程语言中可以实现这一要求，但实现起来有点笨拙。例如，当给定要在 C 中加密的字符串时，必须使用移位运算符来访问特定的位，可以将位表示为其字符阵列以改善对位的访问，但是当需要位实际表示为数字时，例如 DES 中的索引或整数，这种表示又会引入新的困难。

而且，密码专家对 C 语言往往既不精通也不信任，通常密码算法设计好之后，需要找两个程序员背对背的编写代码，除了向其提供算法框架图，还需要花时间列公式、讲解，如果两个程序员给出了不同的结果，就要检查是谁的问题，这是一个非常繁琐的过程。

因此，在密码学领域，迫切需要一种面向加密算法的领域编程语言。

3.1. Cryptol DSL

国际上，由 Galois 公司设计的加密算法编写工具 Cryptol DSL [20] [21]，2008 年正式发布。该语言工具最初由 Galois 为美国国家安全局(NSA)的可信系统研究小组开发，作为指定加密算法的公共标准[21] [22]。

Cryptol 是一种可执行的 DSL，设计人员可以随着设计的发展，逐步测试其程序，其代码转换还可直接应用到硬件电路设计上。

Cryptol 的使用者包括美国军方和美国政府，也被当作一种密码专业的学习工具。该工具集可以生成表示规范和实现的正式模型。其公开的实验表明，用于 AES-256 的 Cryptol 规范生成并在电子密码本模式下运行的算法核心吞吐量超过 16 Gbps。

值得注意的是，Rockwell Collins/Galois 团队可以在 3 个月内，在新硬件上设计，实施，模拟，集成，分析和测试复杂的 CEA，包括 AES-256 和 Galois Counter Mode (GCM)。这样的应用，大大减少了在新平台上完成新密码算法设计的时间。

Cryptol 实现算法类似于数学的规范，它比在通用语言里的实现过程更加的严密。如果将一种规范编写进 Cryptol 里，程序员就可以为自己的程序生成测试矢量、证明定理、验证等效，甚至能生成代码。

与学习任何新的语言范例一样，最初使用 Cryptol 代码工作是困难的，但是一旦它变得舒适，它感觉比 C 好。它的实用性变得越来越明显，代码部分需要单个位操作，例如位串的排列。然而，正如预期的功能语言一样，这种语言由于速度权衡和语言结构的不透明性而主要用于研究或验证其他实现。

为了构建这类算法的更易理解的表示，Galois 公司开发附带了一个工具套件，可用于验证实现用 C 语言编写的算法，如 C 语言。要明确的是，这种语言背后的意图不是用作密码算法准备就绪后的实现，而是作为一种替代方案，提供代码的清晰度以及可证明的功能实现。在为这些类型的密码算法进行原型设计，验证和创建参考的时候，这样的系统将是有用的，这与用于生产用途的其他 DSL(例如 Tensorflow [23])有所不同。

3.2. PCL

PCL 是一款国内自主研发的密码领域专用语言[24]。PCL 的研发始于 Cryptol 发布之初，是一款二维图示化密码算法设计的工具软件，其图形化编辑在 Visio 平台上完成，通过绘制图形而不是编写代码的方式描述密码算法。

按照 DSL 的分类方法，PCL 语言属于“非文本的内部 DSL”。“非文本”是指该语言使用了图形化表述，“内部 DSL”是从一种宿主语言构建而来而不是从零开始构建的语言。

PCL 使用 Cryptol 语言为宿主语言，密码领域需要的解决方案模型，即元语言抽象阶段由 Cryptol 语言完成，图形化编辑则在 Visio 上开发完成。

PCL 设计了从图形到 Cryptol 语言的转换机制：数据流动控制、数据类型匹配、环节代码生成、页面代码封装、回馈数据处理、迭代机制实现等，其正确性建立在 Cryptol 是纯函数式的语言结构上。而编码环节(无论是元环节或复合环节)是模具在 Cryptol 语句集合上的映射，正确性由 Cryptol 语言语法结构的合理性保障。

以 PCL 语言为工具的算法设计流程如图 1。

PCL 还设计了 Cryptol 语言的运行控制机制。把图形转换到 Cryptol 语言之后，控制机制能够启动或中断 Cryptol 的运行，定位每一个中间数据点，能够控制抽取数据、标识数据和存档数据。

在 PCL 平台上，密码算法设计者可以勾勒算法框架、修改算法逻辑、设计或调整变换环节，无需程序员参与模拟即可实时获得实验数据，特别适用于独立不间断的思维过程，以及用于设计或分析时同行之间的交流；对于密码算法的学习者或分析者来说，通过算法逻辑结构图的直观表达能够洞悉设计者的基本思路，对密码算法进行编辑和重构。

PCL 设计有编辑授权机制。鉴于密码算法对数据和变换的敏感性，任何对算法的实质修改必须由授权用户进行，拒绝任何非法修改，合法修改信息会以数字签名的方式记录在算法的页面属性中，用于安

全审计。

在 PCL 之前，包括 Cryptol 在内，没有任何语言可以这样直接描述密码算法的逻辑结构。密码专家们只能以示意图或草图配上必要的数学表达式和文字说明来描述。然而计算机是不能理解示意图这样的表达形式，算法的具体实现只能交给程序员用 C 语言去完成。实际上密码专家对 C 语言既不精通也不信任，往往要找两个程序员背对背的编写代码，除了提供算法框架图，还有花时间列公式、讲解，如果两个程序员给出了不同的结果，就要检查是谁的问题，这是一个十分繁琐的过程。

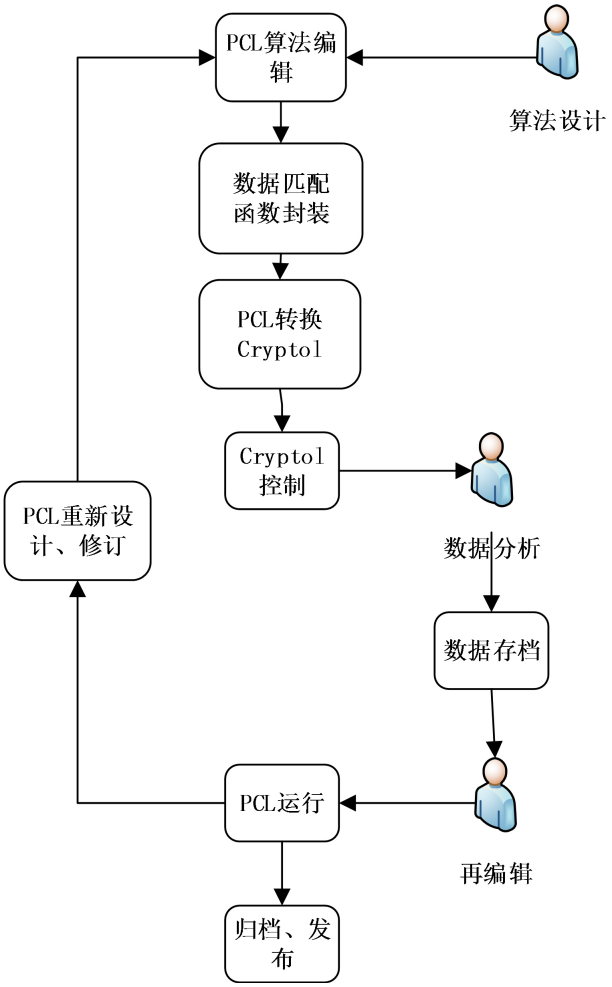

Figure 1. Algorithm design flow using PCL language as a tool
图 1. 以 PCL 语言为工具的算法设计流程

PCL 的基本特点是图形化描述密码算法的逻辑结构。在密码算法框架结构层面上，能满足对密码算法复杂逻辑结构的分层描述。各类密码变换，可以使用特定的模具描述，展现在逻辑图上。

逻辑结构图是密码算法最具代表性的特征，例如 DES 的 Feistel 结构、IDEA 的乘加结构，都以图的形式直观描述数据的变换过程。在密码算法设计者构思一个密码算法时一定会先确定算法的逻辑结构；而算法的分析者在了解一个密码算法时也一定从逻辑结构开始，密码专家的在交流现场是，面对的一定也是算法的逻辑结构图而不是描述密码算法的程序语句序列。

模具是 PCL 从密码变换的繁杂运算中抽象出来的语义模型，模具是密码算法描述的基本单位，其多态性只表现运算的本质而隐藏了运算细节(如算子规模、数据跨界、变量管理等)，使得复杂的密码变换呈

现出极其简约符号形式，和逻辑结构描述连贯构成完备的语言整体。

PCL 建立在 Visio 平台上，其编辑界面分模具栏、编辑栏和属性栏三部分，如图 2。

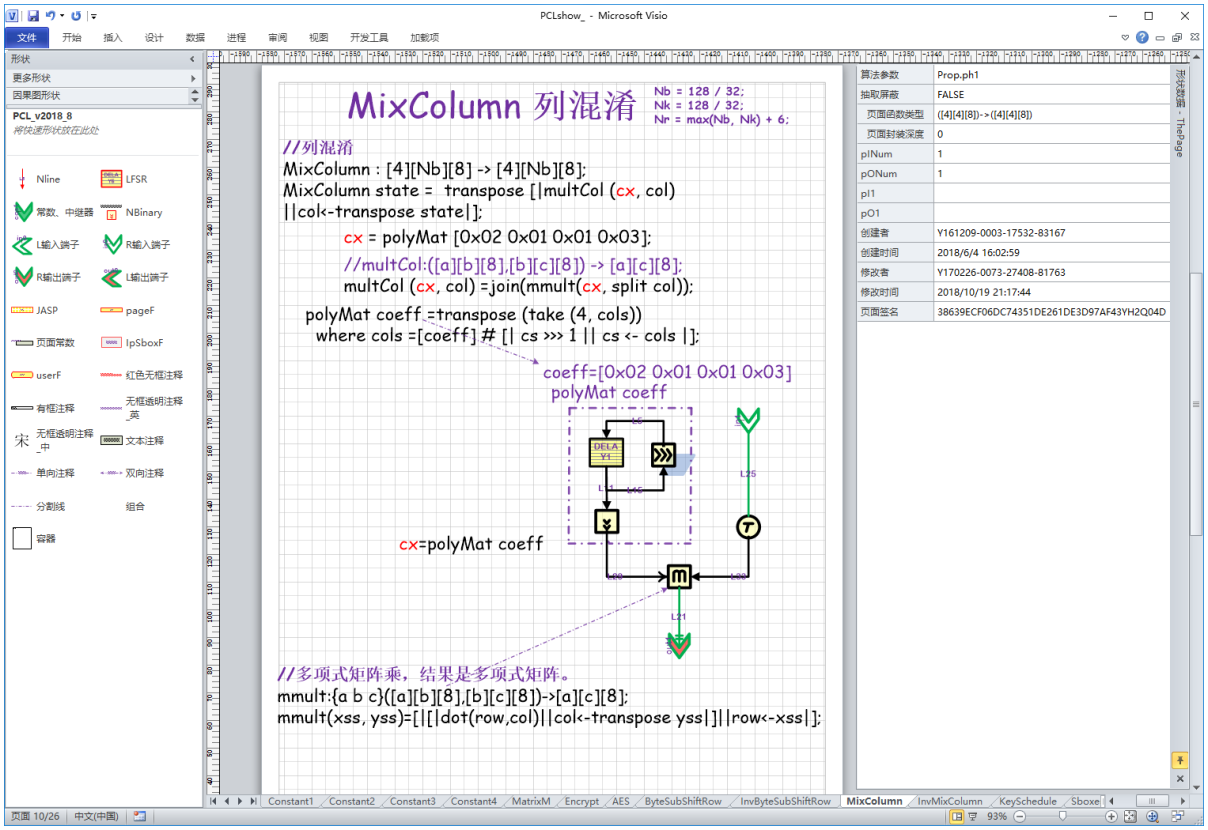

Figure 2. PCL editing interface
图 2. PCL 编辑界面

在 PCL 中，模具有 6 组(分别是 binary、JASP、LFSR、PageF、Line、SBoxF)，以及输入输出端子。其中，binary 涵盖了所有常用的二元变换函数；JASP 能将多个输入数据合成一个大规模数据(join/append)，可以把一个数据等分为几个小规模数据(split)，可以按照置换表把输入数据置换输出(permute)，还可以把以上 3 种操作进行组合运用，构造更复杂的函数；LFSR 可工作在二元域或 2^n 元域上，支持乘法电路和除法电路，二元域互反多项式转换。再通过属性或设置数据，可自动选择适合的工作方式，有：规则移存器、钟控移存器(走停、多步)、带输入移存器、钟控 + 带输入移存器、作为反馈多项式的特例，也可做成延时器；pageF 把一个页面包装为一个环节，使其可以以步进方式或序列方式运行；Line 是连接任意数据(如 Bit 向量，向量的向量、向量的矩阵，矩阵的矩阵)通过的桥梁；SboxF 支持表达式和真值表两种等价表达。

PCL 编辑过程：用户把模具从模具栏中拖入编辑界面，在属性栏中设定模具属性使之成为一个编码环节，用连接线(模具的一种)把编码环节连接起来，构成算法(子算法)：

PCL 运行过程：在编辑界面上用鼠标点击进入运行状态，界面见图 3。运行状态下，用鼠标移到可以查看每条数据线上数据类型，可以点击数据线抽取数据，数据抽取后显示在浮动窗中。对每一条数据线都可以实时查看。如果必要，点击调试面板上运行算法按钮，可以把该数据抽取做为数据文件存储。数据文件中记录了该数据的算法名称、抽取位置、抽取位置、数据规模以及运行平台和操作人员的信息，便于存档。

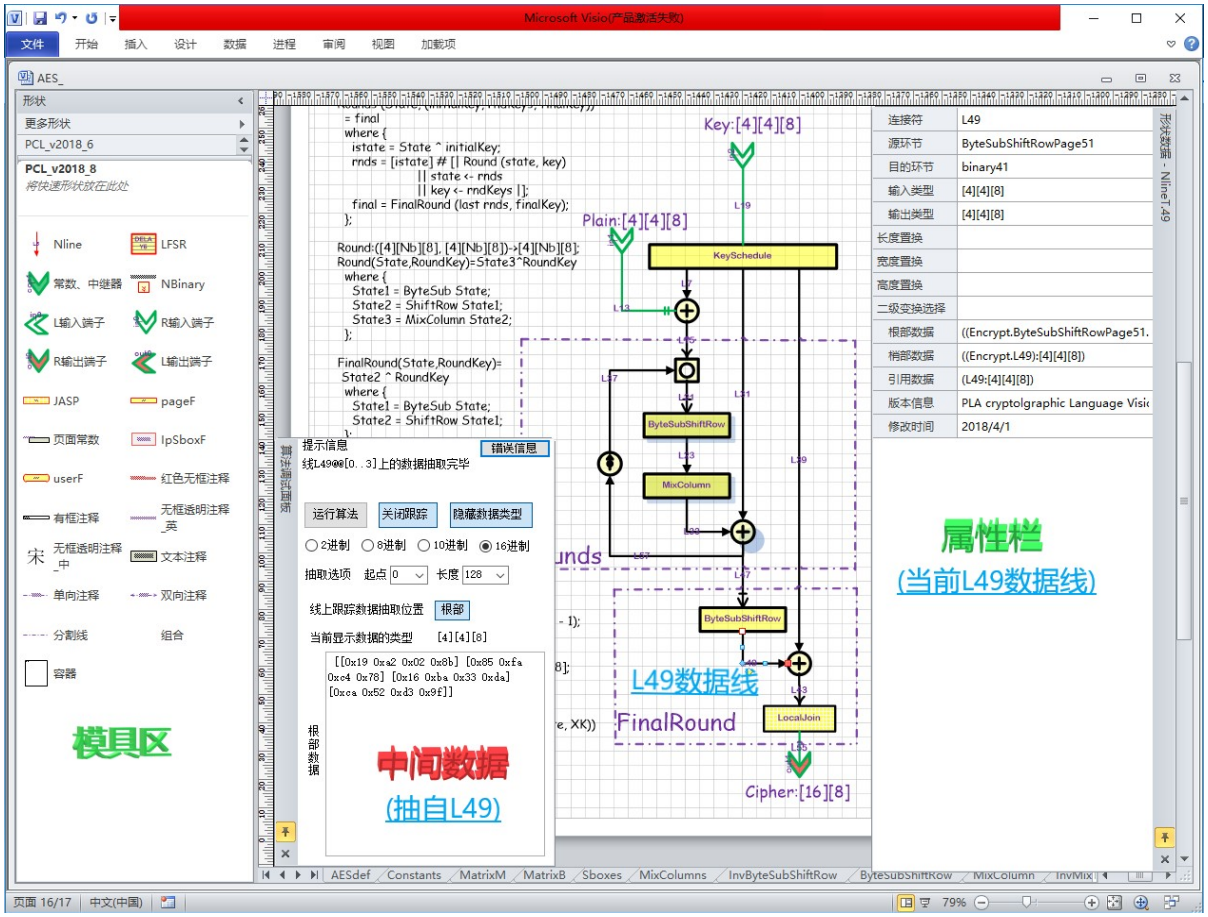

Figure 3. PCL running interface
图 3. PCL 运行界面

4. 结论

本文探索了领域专用编程语言的研究状况，针对密码学领域的专用语言的需求和应用进行了分析，展示了一款国内首次研发的密码学领域专用语言 PCL。密码算法设计者可以利用 PCL 勾勒算法框架、修改算法逻辑、设计或调整算法变换环节。通过算法逻辑结构图的直观表达能够展示、洞悉设计者的基本思路，并对密码算法进行编辑和重构。

在 PCL 之前，包括 Cryptol 在内，没有任何语言可以这样直接描述密码算法的逻辑结构。

基金项目

公安部重点实验室金(C18605)；863 面上(2009AA01Z435)，863 重大(2009AA01Z403)；自然科学基金青年(60272091)；自然科学基金面上(60373109)；重点实验室开放基金(20120316)资助。

参考文献

[1] Domain-Specific Language. https://en.wikipedia.org/wiki/Domain-specific_language

[2] Fowler, M. (2018) Domain Specific Languages. http://adv-r.had.co.nz/dsl.html

[3] Van Deursen, A. and Klint, P. (2018) Domain-Specific Language Design Requires Feature Descriptions. https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=69420

[4] Klint, P., Van Der Storm, T. and Vinju, J.J. (2018) On the Impact of DSL Tools on the Maintainability.

http://dl.acm.org/ft_gateway.cfm?id=1868291&type=pdf
[5] (2018) ANTLR. https://en.wikipedia.org/wiki/ANTLR
[6] (2018) OMeta. https://en.wikipedia.org/wiki/OMeta
[7] (2018) COBOL. https://en.wikipedia.org/wiki/COBOL
[8] (2018) Fortran. https://en.wikipedia.org/wiki/Fortran
[9] (2018) Lisp. https://en.wikipedia.org/wiki/Lisp
[10] (2018) YACC. https://en.wikipedia.org/wiki/Yacc
[11] (2018) SDL. https://en.wikipedia.org/wiki/SDL
[12] (2018) Excel. https://en.wikipedia.org/wiki/Excel
[13] (2018) Ruby. https://www.ruby-lang.org/en/
[14] (2018) Scala. https://scala-lang.org/
[15] van den Brand, M.G.J. (2018) Compiling Language Definitions: The ASF + SDF Compiler. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.5167
[16] Stratego. https://link.springer.com/chapter/10.1007%2F3-540-45127-7_27, 2018-11.
[17] TXL. https://en.wikipedia.org/wiki/TXL_(programming_language), 2018-11.
[18] Hedin, G. and Magnusson, E. (2003) JastAdd—An Aspect-Oriented Compiler Construction System. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.8197, 2018-11.
[19] IEEE Standard for Software Maintenance. https://ieeexplore.ieee.org/document/720567, 2018-11.
[20] Cryptol The Language of Cryptography. https://cryptol.net, 2018-09.
[21] GaloisInc/Cryptol. https://github.com/GaloisInc/cryptol/releases/tag/2.6.0, 2018-10.
[22] Agosta, G. and Pelosi, G. A Domain Specific Language for Cryptography. http://home.deib.polimi.it/agosta/lib/exe/fetch.php?id=gpa%3Acv&cache=cache&media=gpa:agosta07fdl.pdf, 2018-11.
[23] TensorFlow. https://en.wikipedia.org/wiki/TensorFlow, 2018-11.
[24] 联系作者获得语言安装包. h20031795@vip.163.com