面向安全协议的虚拟化可编程数据平面

祝现威* 常朝稳 秦 晰 左志斌
(解放军战略支援部队信息工程大学密码工程学院 郑州 450004)

摘 要：随着网络安全技术的发展，越来越多网络安全协议出现，因此需要网络转发设备对网络安全协议提供支持。可编程数据平面由于其协议的无关性，能够实现安全协议的快速部署。但当前可编程数据平面存在包头多次解析、独占数据平面和密码算法实现难的问题。针对上述问题，该文提出一种面向安全协议的虚拟化可编程数据平面(VCP4)，其通过引入描述头降低包头解析次数，提高包头解析效率。使用控制流队列生成器和动态映射表实现可编程数据平面的虚拟化，实现多租户下数据平面的隔离，解决独占数据平面问题。在VCP4的语言编译器中添加密码算法原语，实现密码算法可重用。最后针对VCP4资源利用率，虚拟化性能和安全协议性能进行实验评估，结果显示在实现功能的基础上带来较小的性能损失，且能降低50%的代码量。

关键词：密码算法原语；可编程数据平面；描述头；虚拟化；控制流队列生成器；动态映射表
中图分类号：TN918; TO391 **文献标识码**：A **文章编号**：1009-5896(2021)01-0226-08
DOI: 10.11999/JEIT190720

VCP4: Virtualization of the Programmable Data Plane for Security Protocol

ZHU Xianwei CHANG Chaowen QIN Xi ZUO Zhibin
(*School of Crpytographic Engineering, PLA Strategic Support Force Information Engineering University, Zhengzhou* 450004, *China*)

Abstract: With the development of network security technology, network security protocol emerges one by one, which requires functional support from network forwarding devices. Due to the independence of protocols, the programmable data plane enables rapid deployment of security protocols. However, the current programmable data plane has the problem that the header is parsed multiple times, the exclusive data plane and the cryptographic algorithm are difficult to implement. In view of the above problems, VCP4(Virtualization Cryptogram P4) as a virtualized programmable data plane for security protocols is proposed, which reduces the number of parsing times and improves the header parsing efficiency by introducing a description header. The control flow queue generator and the dynamic mapping table are used to achieve the virtualization of the programmable data plane, thereby realizing the isolation of the data plane under the multi-tenant and solving the problem of the exclusive data plane. A cryptographic algorithm primitive is added to the VCP4 language compiler to implement a cryptographic algorithm that can be reused. Finally, the VCP4 resource utilization, virtualization performance and security protocol performance are evaluated. The results show that the implementation of VCP4 brings less performance loss, and the code amount can be reduced by 50%.

Key words: Cryptographic algorithm primitive; Programmable data plane; Description header; Virtualization; Control flow queue generator; Dynamic mapping table

1 引言

软件定义网络[1](Software-Defined Network，SDN)将控制平面和数据平面进行解耦，使得网络中引入新的网络协议和功能更加容易。但是目前OpenFlow[2]协议定义的协议类型和字段都仅限于网络前4层中一些常用的协议和字段。但是许多安全协议需要添加新的匹配字段和安全功能，所以为了能够兼容多种网络安全协议，需要一种可编程数据平面，能够使网络管理者自定义匹配字段和转发动作。研究表明将安全功能部署在数据平面能够实

收稿日期：2019-09-17；改回日期：2020-08-30；网络出版：2020-09-16
*通信作者： 祝现威 1056670972@qq.com
基金项目：国家自然科学基金(61572517)
Foundation Item: The National Natural Science Foundation of China (61572517)

现更高的性能，并且能够节省控制器CPU资源。因此越来越多的网络安全协议将安全功能部署在数据平面上。但是目前网络安全协议部署在可编程数据平面仍存在以下问题：(1)包头解析。由于安全协议需要在数据包中插入密文或者签名，需要自定义匹配字段，所以在传统的包头解析中需要分层多次提交，产生较大的延迟。(2)独占的可编程数据平面。当前可编程数据平面是独占设备的，一旦安全功能部署之后，该设备只能运行一个可编程数据平面程序。(3)密码算法编程可重构。在同一网络安全协议下有多种密码算法。因此程序员需要实现不同的密码算法来满足不同的安全需求，并且当前P4并没有提供密码算法方面的编程原语，以上原因增加了安全协议编程难度。

针对以上问题本文提出了一种面向安全协议的虚拟化SDN可编程数据平面。本文主要贡献如下：

(1) 提出一种快速的包头解析器，其为数据包添加一个密码协议描述头来降低提交次数，实现包头的快速解析，在实现自定义字段解析的基础上提高解析效率。

(2) 提出一种基于P4的虚拟化可编程数据平面VCP4，通过对P4进行虚拟化，能够在同一个设备上虚拟出多个隔离的数据平面。实现多租户下的同一设备的可编程数据平面隔离。

(3) 使用P4语言编写部分密码算法相关运算，将其作为密码算法所需的基本运算单元，通过代码的可重用将单元组成对应密码算法原语，实现内部资源可以动态的编排组合。

为了支持不同网络功能的快速部署，首先需要网络设备能够适应不同的协议，能够对包头进行灵活的解析，曹作伟等人[3]提出协议无感知转发交换机，实现协议无关性解析。但是上述解析只支持少量数据包的解析，无法满足大量数据处理。Chole等人[4]提出了一种弹性包头解析器，通过存储包头指针实现多匹配域解析，但是其只能实现4个匹配域的解析，灵活性欠佳。斯坦福大学的Bosshart等人[5]为了完全实现网络从转发到匹配字段的全软件定义，提出了P4(Programming Protocol independent Packet Processor)，受到业界广泛关注，称为OpenFlow 2.0。

在P4虚拟化方面，Hancock等人[6]首次提出了P4虚拟化这一概念，并实现虚拟化可编程平面Hyper4，其在硬件和P4程序之间添加一个类似Hypervisor的管理器实现P4的虚拟化。Hyper4使用了大量的重提交操作，造成效率降低。针对以上问题Zhang等人[7]提出一种全虚拟化的数据可编程平面HyperVDP。虽然实现了虚拟化功能，但无法实现功能和代码的可重用，所以其资源消耗较高。

在功能可重构方面，Zhou等人[8]将P4部分项目进行分解，并通过一个模块化编程管理器ClickP4实现网络功能的动态编排和代码的可重用。季新生等人[9]将可编程数据平面与密码学结合，实现网络认证转发。

2　VCP4总体架构

2.1　总体架构

VCP4的总体架构如图1所示。VCP4作为管理程序在P4编译器与P4硬件之间，其由密码算法编译器、包头解析器、控制流队列生成器和动态映射表4部分组成，能够动态地创建虚拟实例，且独立运行单个P4功能程序。密码算法编译器主要对P4程序以及本文构造的密码运算单元生成的原语进行解析。包头解析器对安全协议包头进行快速解析，并且能够识别用户自定义匹配字段。控制流队列生成器将P4中多种控制流进行单一表示，将控制流中任意有向无环图(Directed Acyclic Graph, DAG)结构转换成线性DAG结构。动态映射表将设备物理资源进行动态映射，实现可编程平面的虚拟化。接下来对各模块进行详细说明。

2.2　包头解析

安全协议往往需要自定义匹配字段，需要对字段灵活解析，所以包头解析器是实现网络安全协议自定义匹配字段的关键，其根据字段偏移量对包头进行分层解析。通常包头解析器通过重复提交实现包头文件的分层解析，最终得到所有的匹配项。在数据包重提交过程中会产生较大延迟，提交次数越多延迟越大。因此为了实现包头的快速解析，需要减少提交次数。

为了降低性能损失，实现快速解析本文在收到

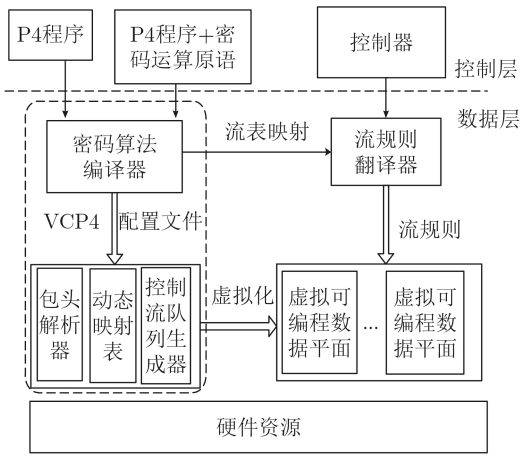


图 1 VCP4总体架构

的数据包头前封装了一个4 Byte的密码协议描述头。它包含虚拟id(8 bit)、总长度(16 bit)和填充位(8 bit)。解析器可以通过一次提交获取真实包头的长度，该方法中原包头作为一个整体，不需要进行分层解析，来获取自定义字段长度。

本文通过解析+匹配表的方式为每个数据包添加描述头。首先需要为每个虚拟数据平面配置虚拟id和其接收数据流的源MAC地址。当接受到数据流后，通过Parser中current offset指针获取包头长度和源MAC地址，根据MAC地址分配虚拟id。通过流表匹配在包头处添加描述头，随后将添加过描述头的包头提交给P4包头解析器。由于数据缓存的存在，一个数据包会多次触发同一原语，造成缓存锁死，其他数据包无法进入缓存进行处理，因此需要进行数据流隔离保证数据平面的可执行性。为此本文在缓存前添加一个流量阀，当某一虚拟id流量过大时，屏蔽该虚拟id的流量进入缓存，将流量降至阈值以下，因此进入包头解析器前过程将在3.1节中叙述。

2.3 控制流队列生成器

P4的控制流由一组阶段(stage)和部分布尔函数组成。Stage可以根据布尔表达式的结果跳转到另一个stage。stage间的跳转实质上是一个有向无环图(DAG), stage表示每个点，stage分支表示点与点之间的边，其值为布尔函数。为了实现控制流的虚拟化，本文需要将控制流进行标准化，即对match-action进行解耦。控制流主要存在stage跳转、匹配和动作执行3个状态，因此本文使用3个流处理过程来描述控制流，其分别为跳转处理过程、匹配处理过程和动作执行处理过程，如图2所示。

跳转处理过程主要用于模拟决定stage跳转的布尔条件，其通过4个流表实现，其流表匹配结果分别为进入匹配处理过程，进入动作执行过程，跳过后续处理和跳入下一个stage。匹配处理过程主要用于字段匹配和动作执行映射。在字段匹配方面，本文将P4中标准match-action表的匹配字段划分为3个类型：包头、标准元数据和用户自定义元数据。动作执行处理过程主要包含所有需要执行的动作，并按匹配处理阶段的动作执行映射顺序执行所有动作。动作执行处理过程中除了P4定义基本动作，本文还根据密码算法添加部分运算作为元动作，如循环移位、欧拉函数计算等。

通过对match-action进行解耦，本文将控制流生成标准化格式，利用文献[10]的方法将这些任意结构的DAG图生成线性结构DAG。首先为每个stage分配一个固定id用于节点编号。

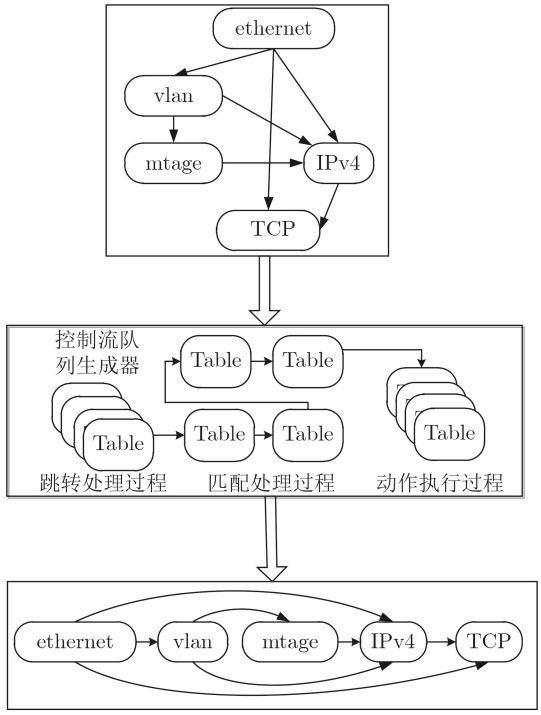


图 2 Match-action解耦

2.4 动态映射表

流队列生成器将任意状态控制流生成有序的线性序列。然而P4中的控制流可以包含任意数量的stage，无法对其数量进行预测，但是硬件资源有限，当多个控制流映射到同一设备上时，容易产生状态溢出。为了解决资源不足这一问题，本文借鉴了虚拟机映射缓存机制[11]和HyperVDP[7]中stage映射方法。因此本文为VCP4引入动态映射表，通过动态映射实现设备的逻辑扩展，使其能够支持不同数量的stage。其结构如图3所示。

首先，由于控制流队列生成器通过对match-action解耦使stage能够独立，因此本文根据设备构造一组连续的stage动态映射表，将解耦后的stage映射到动态映射表中，其类似虚拟内存和物理内存的关系。其次，由于每个stage都拥有一个固定id，因此本文对id和映射表总数进行取模运算来确定stage对应的动态映射表的位置。但是由于P4为了防止内部循环，其stage只能向后产生分支，此外由于不同if-else分支上的stage的绑定条件不相关，所以可以在同一个动态映射表中存在多个stage。当一个控制流拥有的stage多于动态映射表的总数时，VCP4通过重提交来恢复处理。

3 基于P4的密码算法的原语编程

基于P4的密码算法的原语编程主要通过密码算法编译器实现的，其功能由两部分组成：(1)对虚拟可编程平面进行配置；(2)扩展P4编译器，使

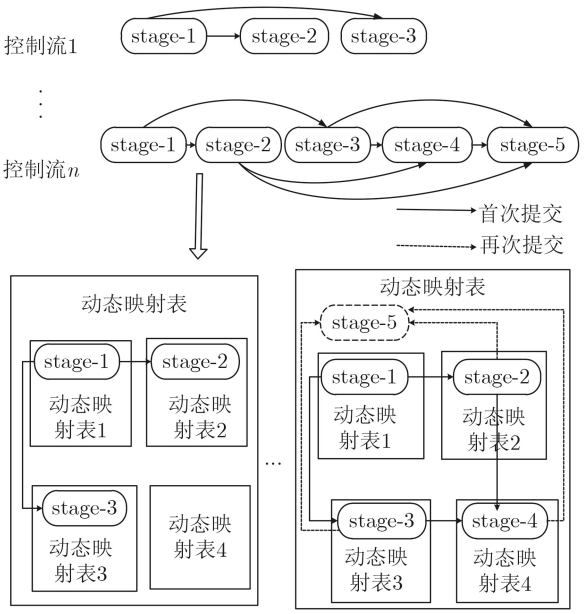


图 3 动态映射表

其不仅支持P4原语还有能够支持P4后端编译的密码算法原语。

3.1 虚拟可编程平面配置

密码算法编译器获取一个P4程序或者P4+密码运算原语程序后，编译程序并为这个程序分配stage动态映射表。随后对虚拟可编程平面进行实例化，其中包括程序id，stage id，填充表项等。每个数据包在描述头位置都会分配一个虚拟id，表示其对应的虚拟可编程平面。当数据包被对应的虚拟可编程平面处理时，通过密码算法编译器为数据包动态地分配程序id和stage id。在动态映射表匹配处理过程中，流表通过与数据包分配的程序id和stage id实现动作执行和阶段跳转。由于使用id对数据包进行划分，以此实现了虚拟可编程的平面的隔离。

VCP4处理控制流过程分为3部分，如图4。首先对数据包头进行快速解析，当数据包进入包头解析时首先添加描述头，并在描述头位置都会分配一个虚拟id。随后将数据包送入缓存，在缓存处通过流量阀检查同一虚拟id数据包流量是否超过阈值，如果小于阈值则通过包头快速解析器提取包头信息。随后解耦match-action过程，通过动态映射表将每个控制流的stage分配到对应的stage容器中。该过程的开头和结尾分别有2个配置表，第1个为数据包匹配的起始地址，第2个是将匹配域超出容器数量的数据包进行重新提交。

密码算法编译器为程序员提供灵活的程序管理方法，首先可以对stage容器进行动态的组合实现复杂的网络功能。其次由于虚拟可编程平面间的隔离性，每个虚拟可编程平面可以表示为一个功能路

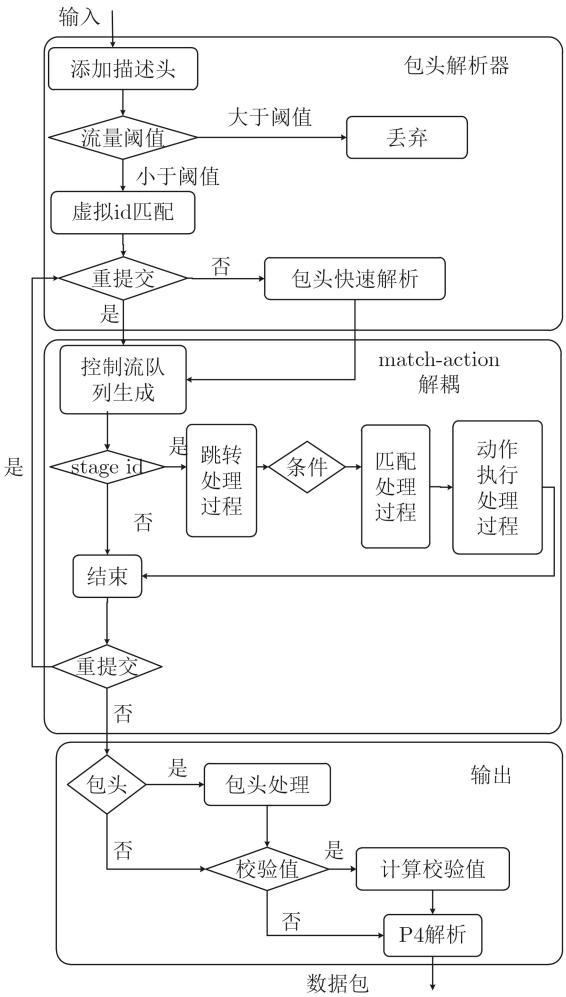


图 4 VCP4中程序处理过程

由，可以通过虚拟平面进行组合完成复杂的网络任务和网络安全协议。

3.2 密码算法原语编程

本文在P4编译器的基础上添加了一个非侵入式的类宏框架，该框架为可编程数据平面提供密码算法的抽象。本文在switch.p4[12]的基础上添加了12个密码算法原语，这些算法原语支持P4-14版本。

表1为密码算法原语列表。表中的密码算法原语分为两种：(1)在单个可编程数据平面的密码算法原语；(2)多个可编程数据平面的状态同步原语。本文将从这两个方面进行介绍。

在单个可编程数据平面密码原语中包括系统功能型和密码算法型两种，系统功能源语主要应用于系统状态编程，无法直接对数值进行运算，需要利用指针进行赋值。前者有@VCP4_for, @VCP4_minmax, @VCP4_cmp，后者主要为密码算法提供运算组合元素，@VCP4_gcd, @VCP4_power, @VCP4_hash, @VCP4_ROL, @VCP4_byte, @VCP4_S, @VCP4_GF。由于篇幅有限，本文在每种密码算法原语中挑选1种进行举例。

表 1 密码算法原语列表

功能	引用注释	描述
欧几里得算法	@VCP4_gcd	求最大公约数
蒙哥马利算法	@VCP4_power	对索引变量进行模幂运算
循环	@VCP4_for	迭代索引变量
MD5	@VCP4_hash	对索引变量进行MD5运算
极大极小值	@VCP4_minmax	从输入列表中选择最大或最小值决定动作执行
循环移位	@VCP4_ROL	对索引变量进行循环移位
条件测试	@VCP4_cmp	对索引变量进行条件测试
比特置换	@VCP4_byte	对索引变量进行置换，实现扩散
S-盒运算	@VCP4_S	对二进制数进行盒运算
有限域乘法	@VCP4_GF	对索引变量进行有限域乘法运算
同步多播	@VCP4_sync	多个可编程平面共享状态
同步单播	@VCP4_echo	与目标可编程平面共享状态

首先本文先介绍系统功能型原语@VCP4_for，当使用不同索引值重复同一原语时可以使用该原语表示。例如构造表头，初始化参数，构造多个相同的流表或者执行相同action。

$$\left.\begin{array}{l}\text{Syntax}\\ \text{@VCP4\_for}(<\text{iterator\_var}>)\\ (<\text{start}>,<\text{end}>,<\text{step}>)\end{array}\right\} \tag{1}$$

式(1)表示对索引变量进行循环赋值。

接下来本文对密码算法型原语@VCP4_ROL进行详细分析。用于实现循环移位，属于DES的基础运算。

$$\left.\begin{array}{l}\text{Syntax}\\ \text{@VCP4\_ROL}(<\text{relop}>)\\ \text{@VCP4\_case}(<\text{custom\_header\_field}>,\\ <\text{field\_value}>,<\text{field\_longth}>);\end{array}\right\} \tag{2}$$

将字节长度为8的自定义字段进行循环左移2位。

如果安全协议需要多个可编程数据平面共同完成，这些数据平面间需要进行频繁的通信和状态同步。因此，本文设想两种多可编程数据平面间的通信方式：同步多播，在虚拟化可编程平面中的同步多播与传统意义上的多播不同，其为一个实体交换机上的虚拟可编程平面向另一个实体交换机上的虚拟可编程数据平面集合发送状态或请求；同步单播，即在同一个虚拟可编程平面集合中不同虚拟可编程平面通信。

同步多播：本文使用VCP4自动生成一组P4代码实现状态变量同步。

$$\left.\begin{array}{l}\text{Syntax}\\ \text{if}(<\text{condition}>)\{\\ \text{@VCP4\_echo}(<\text{custom\_header\_field}>,<\text{field\_value}>)\\ \quad <\text{sync\_var\$1}>\\ \cdots\\ \quad <\text{sync\_var\$}n>\\ \text{@VCP4\_endecho}\\ \}\end{array}\right\} \tag{3}$$

当pkt.time低于特定阈值时，触发时间同步。此时VCP4的密码算法原语会自动生成一个开发人员编写的时间同步操作action。目前默认操作为克隆一个状态同步的数据包。该段action动作在包头指定字段作为同步消息，并且修改目的地址为可编程数据平面集合所在网段。

同步单播：该原语的原理与同步多播类似，唯一不同的是状态信息只发送给单个目的交换机。本文还是以SSH为例，当一个数据平面将状态变量作为请求响应发送给另一个数据平面，这一过程与同步多播类似，唯一不同的是通过在runtime中添加一个返回命令使其只返回给源数据平面。

$$\left.\begin{array}{l}\text{Syntax}\\ \text{if}(<\text{condition}>)\{\\ \text{@VCP4}_{\text{sync}}(<\text{custom\_header\_field}>,<\text{field\_value}>)\\ \quad\quad <\text{sync\_var\$1}>\\ \cdots\\ \quad\quad <\text{sync\_var\$}n>\\ \text{@VCP4\_endsync}\\ \}\end{array}\right\} \tag{4}$$

4 仿真实验

本节实验分从4个方面对VCP4进行评估：(1)包头解析性能，与P4, Hyper4等可编程数据平面的解析器进行比较；(2)在资源利用率方面，将VCP4与现有的虚拟化平面Hyper4及P4进行比较；(3)在虚拟化性能方面，本文将VCP4 DPDK-target与P4

DPDK-target和Openvswitch虚拟化交换机进行对比；(4)安全协议性能方面，本文分别在P4和VCP4 BMv2-target上实现3个安全协议用例来进行比较说明。

4.1 资源利用率

可编程数据平面的主要资源为流表和metadata元数据，因此本文从VCP4中流表数量和虚拟可编程数据平面runtime中流表数量2个方面来评估VCP4的资源利用率。

(1) 由于硬件资源的限制，为了将更多的资源用于虚拟可编程平面的运行，因此VCP4的运行管理应当用尽可能少的流表数量，提高管理效率。因此本文分别在不同数量原语和stage的条件下，将VCP4与Hyper4、P4进行流表数量对比。本文对VCP4与Hyper4下发相同stage的控制流，且VCP4设置4个动态映射表，在此条件下观察实现不同数量原语需要的流表数量，试验重复10次并求平均值。

从图5(a)可知，随着原语数量增加，Hyper4和P4的流表数量快速增加，而VCP4几乎不变。这是由于Hyper4和P4对原语处理主要依靠线性流表处理管道，每条原语都会产生对应的管道，因此Hyper4和P4的原语处理与流表数量呈线性正相关，由于P4不需要进行虚拟化配置所以流表数量低于Hyper4，而VCP4由于利用动态映射表循环处理，没有原语流表处理管道，其主要与动态映射表的数量和配置流表数量有关。

(2) 虚拟可编程数据平面runtime中流表数量表示虚拟可编程数据平面实现功能时产生的流表，越多的流表表示其处理管道越长，带来的处理延迟越大。本文分别通过VCP4, Hyper4和P4分别实现防火墙[13]、交换机[14]、ARP代理[15]和IPsec认证(AH头中使用字段匹配替代密码算法)，结果如图5(b)所示。

在试验中，P4通过2个流表实现交换机功能，分别为源MAC地址匹配表和目的MAC地址匹配表。在VCP4中，需要5个表，1个虚拟可编程数据平面配置表，2个匹配流表，2个动作流表。Hyper4则需要18个流表实现这一功能。在L2层实现防火墙，VCP4只需要2个动态映射表，每个动态映射表包含2个匹配表和动作表，而Hyper4需要多个表对数据包进行解析，再需要至少4个match-action表用于不同数据包的匹配。在实现ARP代理和IPsec时三者的流表数量增加较多，其主要是因为需要进行TTL检查、TTL修改和TTL计算等操作。一般来说Hyper4使用的流表数量至少是VCP4的2倍。

4.2 虚拟化性能

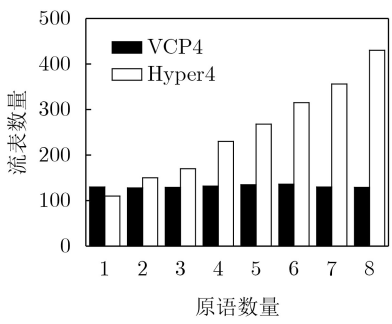
由于VCP4是在DPDK-target基础上实现虚拟化功能，所以本文将VCP4与同样使用DPDK-target的P4和Openvswitch进行对比。分别通过以上3种数据平面转发40 Gbps数据量，通过测量其吞吐量、时延和CPU利用率对其虚拟化能力进行评估。本文分别在3种结构上实现L2层转发。

(1) 吞吐量测试。结果如图6(a)所示。从图中可知对于短数据包，由于VCP4和Openvswitch引入了虚拟化功能，所以吞吐率略低于P4，但当发送的数据包为长数据包时，三者吞吐率一样。VCP4相对于P4吞吐率总体损失约9%。

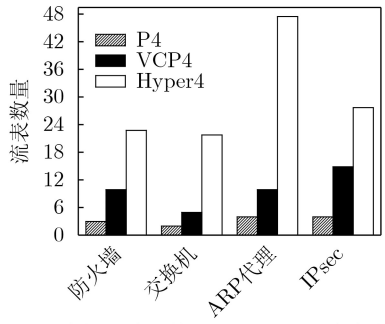
(2) 时延是衡量网络质量和可用性的一项重要指标。本文分别测量了VCP4, P4和Openvswitch平均转发时延，结果如图6(b)。时延随着数据包的长度增加而增加，VCP4时延高于P4和Openvswitch，增加了约17%。因为P4和Openvswitch使用流水线匹配方式，而VCP4由于虚拟化和可编程的需求，使用队列的方式模拟流水线方式，对收到的数据包进行match-action解耦，所以处理时延高于P4和Openvswitch。

4.3 安全协议性能

安全协议性能主要指实现安全协议功能的可编



(a) 不同原语数量下流表数量



(b) 虚拟可编程数据平面runtime中流表数量

图 5 可编程数据平面资源利用率对比

程数据平面的性能和实现安全协议代码量。本文实现6个网络功能，并根据网络功能组合实现CHAP, IPsec(利用Hash验证完整性)和IPsec(利用RSA认证)3个协议。

(1) 由于P4和Hyper4没有密码算法原语，所以本文将接收到的密码信息通过6653端口导入主机操作系统中，利用对应的密码算法库对信息进行验证，将验证结果返回到可编程数据平面中进行处理。本文分别用BMv2-target的吞吐量和时延评估可编程数据平面实现安全协议的性能，结果如图7所示。

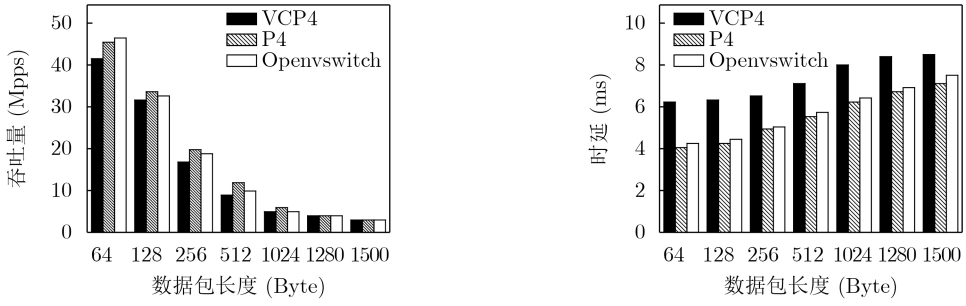
从图7(a)和图7(b)可知，Hyper4的吞吐量远小于P4，虚拟化带来的吞吐量的降低约83%。VCP4吞吐量降低范围是33%～45.6%，其情况远好于Hyper4。这是由BMv2-target处理管道数量和长度决定的，Hyper4需要为每个数据流动作分配一个处理管道，且在解析的过程中需要分层解析，因此其需要多个处理管道，其吞吐率较低。由于VCP4包含密码算法原语，可以直接对包中认证信息进行处理，而P4和Hyper4需要将认证信息导出处理，所以VCP4的时延低于P4和Hyper4。

(2) 从代码量方面分析，本文将VCP4中实现的安全协议代码转化为P4的BMv2-target[16]代码，由于P4没有密码原语，所以本文将P4外部的密码算法库代码量统计在内，结果如图7(c)。从图中可知，由于密码原语简化了密码算法的实现，且部分循环，异步功能实现也进行了简化，大大降低了安全协议实现的代码量，降低至少50%以上。在IPsec框架下，随着密码算法的复杂度增加，P4中密码算法代码增加，但VCP4由于使用密码原语，其代码量变化较小。

5 结束语

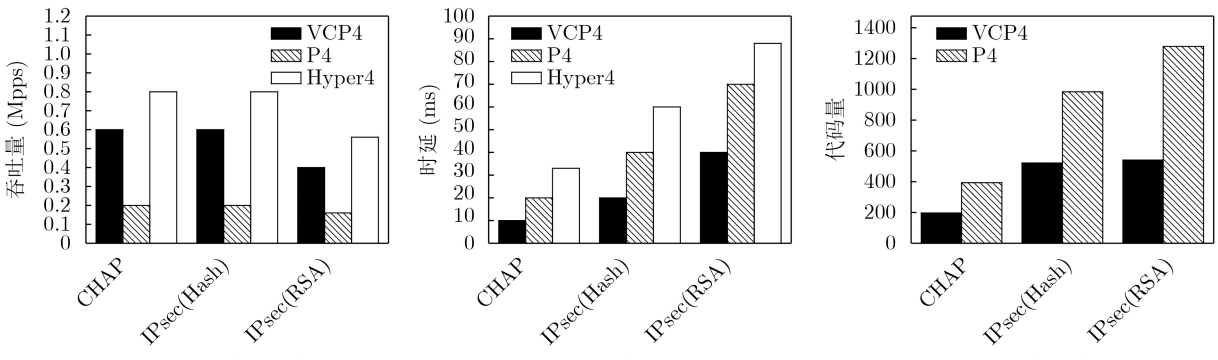
通过在P4的基础上实现虚拟化可编程数据平面VCP4，满足网络安全协议中可自定义字段和数据流的隔离性，其具有以下创新点，首先在P4的基础上引入快速包头解析器、控制流队列生成器和动态映射表技术，在BMv2-target和DPDK-target上构造Hypervisor，以此实现可编程数据平面的虚拟化，通过可编程平面的虚拟化保证网络安全协议的隔离性。其次为了便于网络安全协议的编程，本文在P4语言的基础上添加了密码算法原语，对常用的密码运算进行了封装，这样能够有效降低代码量。

根据VCP4的结构，本文分别从资源利用率、虚拟化性能和安全协议性能3方面对VCP4进行评估。结果显示在DPDK-target方面，VCP4相较于P4吞吐率减少9%时延增加了17%，其在可接受范围内。在BMv2-target方面，VCP4由于密码原语的使用，其时延优于P4，且代码量降低了50%。虽



(a) 吞吐量测试 (b) 时延测试

图 6 可编程数据平面虚拟化性能对比



(a) 吞吐量对比 (b) 时延对比 (c) 代码量对比

图 7 可编程数据平面安全协议性能对比

然VCP4实现虚拟化且降低了安全协议部署的代码量，且带来的性能损耗是可接受的，但是其服务仍是按队列进行的，无法根据网络需求进行动态的分配，无法适应部分时延敏感型业务。

参 考 文 献

[1] MCKEOWN N. Software-defined networking[C]. IEEE International Conference on Computer Communications, Rio de Janeiro, Brazil, 2009: 30–32.

[2] MCKEOWN N. OpenFlow 1.3[EB/OL].https://github.com/CPqD/ofsoftswitch1.3/, 2006.

[3] 曹作伟, 陈晓, 倪宏, 等. 应用于协议无感知转发交换机的流缓存方法[J]. 电子与信息学报, 2018, 40(11): 2772–2778. doi: 10.11999/JEIT180042.
CAO Zuowei, CHEN Xiao, NI Hong, *et al*. Flow caching in protocol oblivious forwarding switches[J]. *Journal of Electronics & Information Technology*, 2018, 40(11): 2772–2778. doi: 10.11999/JEIT180042.

[4] CHOLE S, FINGERHUT A, MA Sha, *et al*. dRMT: Disaggregated programmable switching[C]. ACM Special Interest Group on Data Communication, Los Angeles, USA, 2017: 1–14. doi: 10.1145/3098822.3098823.

[5] BOSSHART P, DALY D, GIBB G, *et al*. Programming protocol-independent packet processors[J]. *ACM SIGCOMM Computer Communication Review*, 2014, 44(3): 87–95. doi: 10.1145/2656877.2656890.

[6] HANCOCK D and VAN DER MERWE J. Hyper4: Using P4 to virtualize the programmable data plane[C]. The 12th International on Conference on Emerging Networking Experiments and Technologies, Irvine, USA, 2016: 35–49. doi: 10.1145/2999572.2999607.

[7] ZHANG Cheng, BI Jun, ZHOU Yu, *et al*. HyperVDP: High-performance virtualization of the programmable data plane[J]. *IEEE Journal on Selected Areas in Communications*, 2019, 37(3): 556–569. doi: 10.1109/JSAC.2019.2894308.

[8] ZHOU Yu and BI Jun. ClickP4: Towards modular programming of P4[C]. SIGCOMM Posters and Demos, Los Angeles, USA, 2017: 100–102. doi: 10.1145/3123878.3132000.

[9] 季新生, 徐水灵, 刘文彦, 等. 一种面向安全的虚拟网络功能动态异构调度方法[J]. 电子与信息学报, 2019, 41(10): 2435–2441. doi: 10.11999/JEIT181130.
JI Xinsheng, XU Shuiling, LIU Wenyan, *et al*. A security-oriented dynamic and heterogeneous scheduling method for virtual network function[J]. *Journal of Electronics & Information Technology*, 2019, 41(10): 2435–2441. doi: 10.11999/JEIT181130.

[10] BANSAL M, MEHLMAN J, KATTI S, *et al*. OpenRadio: A programmable wireless dataplane[C]. The 1st Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 2012: 109–114. doi: 10.1145/2342441.2342464.

[11] NORDAL A Ø, KVALNES Å, PETTERSEN R, *et al*. Streaming as a hypervisor service[C]. The 7th International Workshop on Virtualization Technologies in Distributed Computing, New York, USA: 2013: 33–40. doi: 10.1145/2465829.2465831.

[12] BOSSHART P. P4-bmv2[EB. OL]. https://github.com/p4lang/behavioral-model, 2017.

[13] LIU J, HALLAHAN W, SCHLESINGER C, *et al*. P4v: Practical verification for programmable data planes[C]. 2018 ACM Special Interest Group on Data Communication, Budapest, Hungary, 2018: 490–503. doi: 10.1145/3230543.3230582.

[14] IBANEZ S, BREBNER G, MCKEOWN N, *et al*. The P4-> NetFPGA workflow for line-rate packet processing[C]. 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, USA, 2019: 1–9. doi: 145/3289602.3293924.

[15] MARTINEZ-YELMO I, ALVAREZ-HORCAJO J, BRISO-MONTIANO M, *et al*. ARP-P4: A hybrid Arp-path/p4runtime switch[C]. The 26th IEEE International Conference on Network Protocols, Cambridge, UK, 2018: 438–439. doi: 10.1109/ICNP.2018.00062.

[16] BOSSHART P. Behavioral-model[EB/OL]. https://github.com/p4lang/behavioral-model, 2017.

祝现威：男，1991年生，博士生，研究方向为SDN安全、可编程数据平面.

常朝稳：男，1966年生，教授，研究方向为信息安全、可信计算.

秦　晰：女，1978年生，副教授，研究方向为SDN安全、可信计算.

左志斌：男，1988年生，讲师，研究方向为SDN安全、可编程数据平面.

责任编辑：马秀强