基于身份属性的 SDN 控制转发方法

祝现威，常朝稳，朱智强，秦晰
（信息工程大学密码工程学院，河南 郑州 450001）

摘 要：针对软件定义网络中数据流转发缺少有效的转发验证机制和 OpenFlow 协议匹配字段数量有限的问题，提出了一种基于属性密码的转发控制架构。通过设备属性生成属性标识和属性签名，并将其封装在分组头中。当数据流离开网络时，转发设备对其进行数据验证，确保数据流的有效性。同时，将属性标识作为流表匹配字段，通过属性标识定义网络转发行为，该机制与属性签名验证共同实现细粒度的访问控制。实验结果表明，该系统能有效实现数据流的细粒度的转发认证，且转发粒度高于同类方案。

关键词：软件定义网络；属性标识；转发控制机制；属性签名；访问控制；流表匹配
中图分类号：TP393
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2019232

SDN control and forwarding method based on identity attribute

ZHU Xianwei, CHANG Chaowen, ZHU Zhiqiang , QIN Xi
Department of Cryptogram Engineering, Information Engineering University, Zhengzhou 450001, China

Abstract: Due to the lack of effective data source authentication mechanism and the limited matching fields in software defined networking (SDN), an SDN security control and forwarding method based on identity attribute was proposed. Attribute identification and attribute signature were generated by device attributes and encapsulated in the group header. When the data flow left the network, the data was verified by the forwarding device to ensure the validity of the data flow. At the same time, attribute identification was defined as a match field of flow by the framework, and the network forwarding behavior was defined based on attributeidentification. A fine-grained access control was implemented by the proposed mechanism and attribute-based signature. The proposed mechanism and attribute-based signature implemented a fine-grained access control. Experimental results demonstrate that the method can effectively implement fine-grained forwarding and flow authentication, and the forwarding granularity is higher than that of similar schemes.

Key words: software defined networking, attribute identification, forwarding control mechanism, attribute signature, access control, low matching

1 引言

软件定义网络（SDN, software defined networking）[1]是由斯坦福大学的 CleanSlate 团队提出的一种新的网络结构，其实现了网络数据平面和控制平面的解耦，具有良好的扩展能力和网络编译能力。由于控制平面与数据平面的分离，数据平面只具有数据转发功能，提高了控制器对数据来源进行监控的难度，无法实现端到端的数据认证。因此，目前 SDN 流表转发时还没有提供有效的数据来源认证机制[2]，不能有效防止伪造攻击。

虽然开发人员会为 SDN 提供一部分安全措施，但是安全措施主要集中在控制层[3]，其存在 2 个缺陷：1) 造成控制器臃肿负载增大；2) 安全功能中心化，一旦控制器被攻破，将导致整个网络不安全。并且，由于 OpenFlow 只能根据网络协议的

收稿日期：2019–06–12；**修回日期**：2019–10–23
基金项目：国家自然科学基金资助项目（No.61572517）
Foundation Item: The National Natural Science Foundation of China (No.61572517)

信息特征进行转发控制，不能对网络业务进行划分，因此无法实现网络间细粒度的访问控制，攻击者可以将合法设备作为跳板对 SDN 进行攻击。

数字签名作为数据合法性验证的工具被广泛应用于操作系统和网络中，如 Porras 通过 FortNOX[4]对应用进行加密认证。属性签名[5-6]具有能够在无访问列表的情况下实现细粒度的访问控制，满足不同安全需求，并且只需改变访问属性即可更新访问控制结构等优点，能够满足多样化的安全需求[6]。

因此，本文针对 SDN 用户来源认证方式和细粒度的网络访问控制缺乏、安全功能中心化的问题，将 SDN 与属性签名相结合，提出了一种基于属性标识的 SDN 安全控制转发架构——ACFlow。

ACFlow 架构特点如下。1) 通过设备属性生成属性标识，将数据分组与设备绑定，实现了数据流的标记。2) 利用认证交换机对数据合法性进行认证，确保进入网络的数据分组为合法数据，防止遭受拒绝服务攻击；在数据平面实现安全功能，实现数据端到端的认证，并将数据认证分散在每个数据节点上，降低了控制器的负载，实现了安全功能的去中心化，提高了网络的总体安全性。3) 利用属性签名和基于属性标识的转发实现了基于网络业务的细粒度访问控制和数据来源鉴别。

2 相关工作

目前，针对网络中的非法访问主要是通过下发流表的方式实现的，分为 2 种思路：1) 由控制器对流规则进行认证，保证下发安全的流规则；2) 通过控制器对入网的设备进行认证，并下发入网策略。以上 2 种方法主要是对控制器进行开发。第一种思路主要有基于角色认证的流规则，如 Porras 等[4]提出的控制器 FortNOX，该控制器对下发流规则的应用进行基于角色的认证。FortNOX 对应用认证主要基于 3 个角色：管理员、合法应用和非法应用。这些角色被分配给每个应用并对其进行签名。然后，Porras 等[7]对该架构进行扩展，提出了一种新的安全系统 SEFloodlight，该系统在 Floodlight 控制器基础上添加安全层（SEK, security-enforcement-kernel）实现基于角色的流规则管理，类似的还有 Rosemary[8]和 FRESCO[9]。以上方法虽然能够实现对非法数据流防御，但可能会将不同的安全应用赋予相同的角色，无法实现流的细粒度管理。因此 Wen 等[10]基于控制器的接口提出 18 种权限，并使用 PermOF 系统实现权限分配，与 SEFloodlight[7]、Rosemary[8]和 FRESCO[9]相比实现了流的细粒度管理，但是其不能防止终端处伪造攻击。基于第二种思路，Casado 等[11]提出了 Ethane 架构，通过对主机进行注册认证实现对终端的管理，但是其没有对终端进行角色的划分，无法实现流的细粒度管理。以上 2 种思路都会大量增加控制器的负载，降低控制器下发流表的能力。

对于传统方式的不足，研究者将研究方向从控制器转向了交换机和 SDN 架构。如郑鹏等[12]基于流特征提出 uFlow 技术，将流进行细粒度划分。Ballard 等[13]提出 OpenSAFE，通过 SDN 交换机对网络流量进行重定向并对路由进行检查，但缺乏细粒度的管理。作为补充，Wundsam 等[14]提出 OFRewind 架构，该架构支持多种粒度的管理。Halpern 等[15]提出服务功能链 SFC（service function chaining），利用 SDN 流表进行数据流控制，将数据流以不同维度不同粒度按需求进行划分，根据划分为数据流分配不同功能链和功能链路径，使网络数据可以根据用户属性、业务属性、网络属性等不同层面的需求定制用户数据流经过的服务功能。以上方案虽然可以有效降低控制器的负载，但是安全设备的部署和 SDN 对流量控制的粒度直接制约了数据来源验证的效率。赵志远等[16]提出了虚拟 SDN 映射方法，实现了多控制器条件下的细粒度管理。

现有的 SDN 南向协议主要是 OpenFlow 协议，该协议定义的协议类型和字段都仅限于网络前四层中一些常用的协议和字段，控制粒度有限[17]。但是从 OpenFlow v1.2[18]之后匹配域使用 OXM（OpenFlow extensible match）架构的 TLV（type-length-value）格式，为拓展匹配域的范围提供了可能。例如，Lu 等[19]提出了可编程数据转发引擎 CAFÉ，该引擎通过比特提取器实现了用户对关键字的自定义查找，实现了分组匹配域的自定义扩展，但该引擎并未对分组头解析进行优化，导致解析使性能开销增加。针对分组头解析，Atting 等[20]提出了一种分组头解析算法并为此设计了解析语言 PPL（packet parsing language），除了满足分组头解析外还提高了分组性能。金子晋等[21]根据流表的匹配域对不同业务流进行路径分配，保障用户路由 QoS（quality of service）。

3 攻击场景与模型

常规 SDN 的抽象结构如图 1 所示，包含了如

下元素：源设备、SDN 控制器、为控制器提供合法流规则的服务 s_1、为控制器提供非法流规则的服务 s_2、OpenvSwitch、安全设备（如防火墙、安全网关等）。

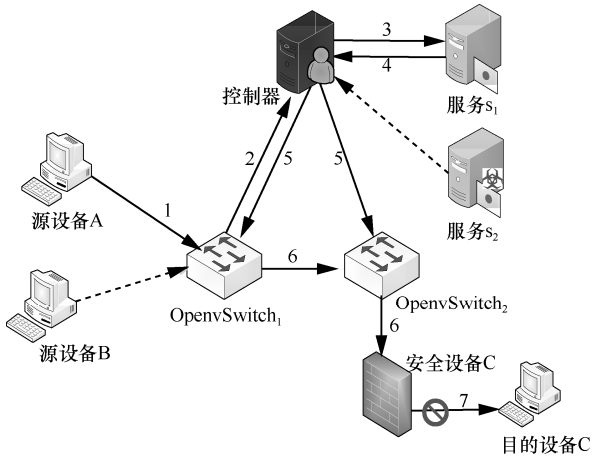


图 1　常规 SDN 抽象结构

假设源设备 A 希望访问目的设备 C，则需要经历以下步骤。

步骤 1　源设备 A 向 $OpenvSwitch_1$ 发送对目的设备 C 的访问请求。

步骤 2　因为没有对源设备 A 请求匹配的流规则，所以 $OpenvSwich_1$ 向控制器请求流规则，等待下一步动作。

步骤 3　控制器向服务 A 申请生成一个新的访问流规则，以此回应 $OpenvSwich_1$ 的访问请求。

步骤 4　服务 A 生成相应的流规则，并将流规则发送给控制器。

步骤 5　一旦控制器收到产生的流规则，首先将该流规则保存在流规则库中，然后将该流规则转发给 $OpenvSwitch_1$ 和 $OpenvSwitch_2$。

步骤 6　新的流规则会匹配源设备 A 中的数据分组，通过 $OpenvSwitch_1$ 和 $OpenvSwitch_2$ 将数据分组转发给安全设备。

步骤 7　安全设备检测发来的数据分组，通过规则判断得出源设备 A 不能与目的设备 C 进行通信。

根据 SDN 结构构造 2 种攻击方式[22-23]，分别通过篡改网络规则和控制网络流量来实现非法访问。还有其他直接攻击场景，如分布式拒绝服务攻击等。下面分别描述篡改网络规则攻击和控制流量攻击过程。

篡改网络规则攻击如图 2 所示。恶意应用构造一个虚假的流表，该虚假流表的流规则使恶意源设备 A 通过 $OpenvSwitch_2$ 直接访问目的设备 C，这样就绕过了防火墙并对目的设备 C 进行扫描攻击。

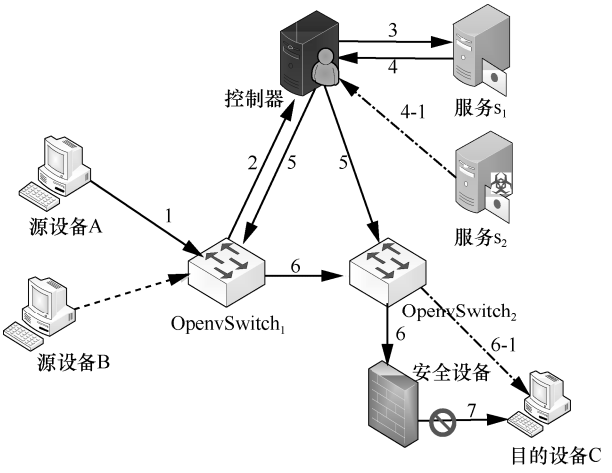


图 2　篡改网络规则攻击

控制流量攻击如图 3 所示。服务 B 伪造 2 个流表。通过第一个流表，将源设备的 IP 地址从恶意源设备 A 修改为合法源设备 B，使用合法源设备 B 访问目的设备 C；然后第二个流表修改数据分组的目的地址，将原本从目的设备 C 到源设备 B 的数据分组目的地址修改为恶意源设备 A，实现恶意源设备 A 对目的设备 C 的扫描攻击。

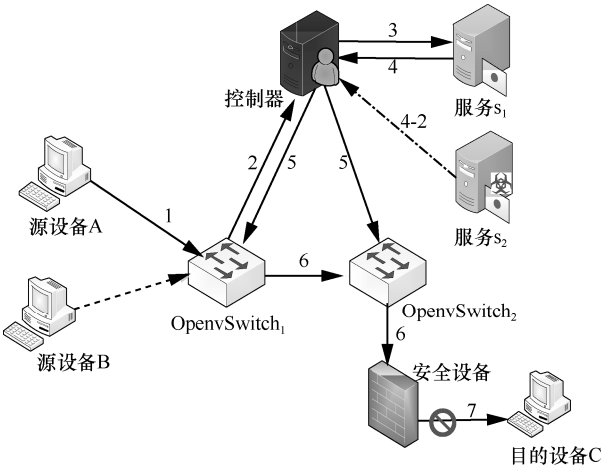


图 3　控制流量攻击

以上 2 个例子说明攻击者可以通过篡改网络规则和控制流量绕过部署在固定路径的防御措施如网络防火墙、安全网关等，实现恶意攻击，使现有的防御方式失效。虽然针对以上问题提出了 FortNOX、SEFloodlight 和 FRESCO 等安全架构，这些架构通过对流规则的合法性认证防止流规则的篡改，但是受到应用角色数量的限制，不同的应用会获得相同的角色权限，因此当存在大量应用时，恶意应用的角色会与合法应用角色产生重叠，获得相同的访问权限，从而出现以上 2 种攻击。上

述问题的实质是数据流的非法访问，因此，通过数据层实现用户合法性认证，阻止非法用户的接入，并且通过属性标识和属性签名实现细粒度的访问控制，以此阻止用户的非法访问。

4　基于属性标识的 SDN 安全控制转发架构

4.1　原理介绍

本文针对上述问题[22-23]提出了 ACFlow 架构，该架构利用属性标识（AID, attribute ID）的身份属性，将属性标识与源设备身份属性进行绑定，并与系统公开参数生成属性私钥，通过属性私钥进行数字签名 Sign 。利用属性标识的密码属性，在网络的出口验证数字签名 Sign ，确保到达数据为合法数据，如果签名不通过，交换机认为该设备为非法设备，将拒绝其接入网络，因此可以保证 SDN 不被恶意用户进行攻击。利用属性标识的标签属性，使属性标识成为转发设备中流表可识别的匹配字段，最后通过多级流表匹配的方式实现基于属性标识定义网络转发行为，扩展网络管控粒度。属性标识的身份属性、密码属性和标签属性共同作用，形成“来源可验证、精细化控制”的安全防护体系架构。

4.2　总体框架

ACFlow 架构的通信模型如图 4 所示，其包含属性标识认证中心、属性标识组件、基于属性标识的控制层和基于属性标识的数据层 4 个部分。

1) 属性标识认证中心。生成系统公开参数，并为目的设备生成访问公开参数。对需要接入的源设备生成基于属性的属性标识，采用属性签名方式，使用属性标识生成属性私钥。

2) 属性标识组件。源/目的设备属性标识管理的核心，以应用的形式安装在主机上。首先，负责为新入网设备生成属性集；其次，从属性认证中心获取属性私钥和属性标识，并对 IP 分组签名，通过修改主机的协议栈实现属性标识和属性签名封装，并且不对主机进行物理扩展；最后，存储该设备的访问结构 T（即设备验证签名所需的属性集，详细说明见 5.1 节）的公开参数，提供给认证交换机的分组鉴别模块读取。

3) 基于属性标识的控制层。主要有以下 2 个功能：通过收集数据层信息，获取网络拓扑结构；生成基于属性标识的流规则实现对数据层的转发控制。控制层主要包括分组解析模块、属性标识失效模块和流规则生成模块，其功能分别为分组解析出属性标识、属性标识有效性鉴别和生成匹配的流规则。

4) 基于属性标识的数据层。主要由认证交换机组成，负责匹配转发收到的 IP 分组。在原有的 SDN 交换机的基础上，增加了属性标识解析和数据来源验证功能，将属性标识扩展为自定义匹配域，可根据属性标识转发 IP 分组，并根据交换机连接主机的访问结构在出口对数据合法性进行验证。

下面以源设备 D_1 访问目的设备 D_2 为例来描述 ACFlow 架构的通信过程。

1) 源设备 D_1 在访问网络前，需要通过本地安装的属性标识组件对其进行初始化，包括上传属性集和目的地址；然后接收生成的属性私钥并生成属性签名，并将属性标识和签名与原分组封装成新的分组。

2) 数据层收到来自 D_1 的分组并对其进行转

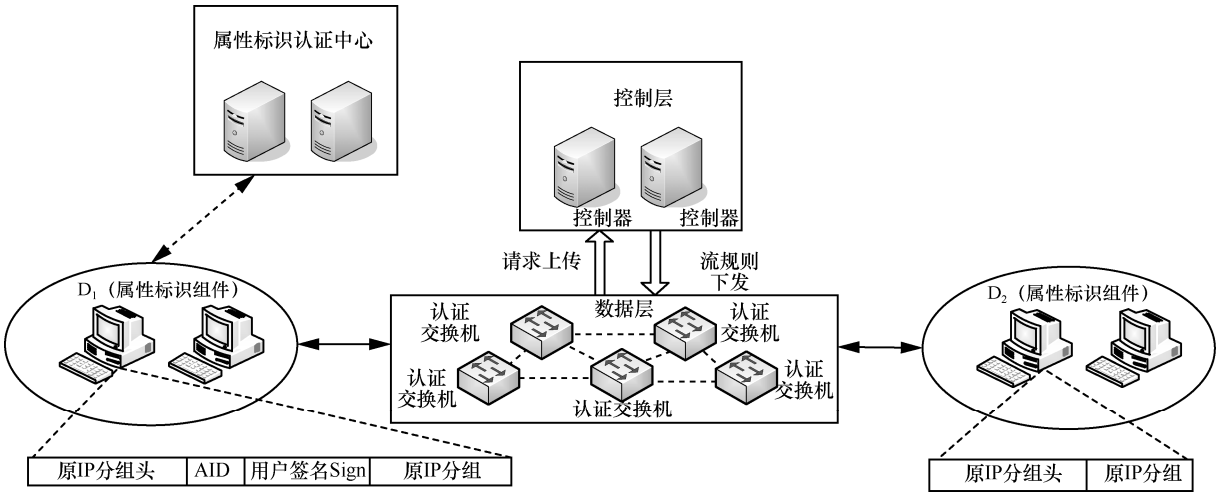


图 4　ACFlow 架构通信模型

发，每次转发前判断路由下一跳是否为目的设备，如果不是则直接进行转发。

3) 当分组根据流表转发到达出口认证交换机（路由下一跳为目的设备）时，对分组合法性进行属性签名验证，如果验证通过则将分组转发给目的设备 D_2，否则将其提交给控制器生成无效流表，控制器将无效流表下发给认证交换机，拒绝非法分组进入网络。

4) D_2 通过本机属性标识组件客户端将收到的合法分组进行解封装生成一般 IP 分组。

4.3 系统结构设计

4.3.1 带属性标识 IP 分组生成

新设备 D_1 接入网络前需要对该设备进行身份定义，传统的方式是对每个设备生成一个唯一的设备号，生成设备号身份对照表，造成系统开销增加。实际上只需要几个属性便可确定用户身份，并且认证时只要用户某些属性满足要求即是合法用户，认证者并不关心签名人的名字、地址等其他信息。本平台从设备所在部门、所有人的角色和从事的业务这三方面定义设备属性，例如小明是 IT 部门的工程师需要对某个主机进行写操作，其属性为 IT部门∧工程师∧write。

假设属性标识组件接受的属性可信没有伪造，通过 Hash 函数 $H:\{0,1\}^* \to Z_q^*$，将属性集中每个属性生成一串 Hash 值，将其生成的布尔函数作为属性标识。属性标识作为 IP 分组进出网络的许可证，负责分组的认证和转发，其位于网络层与传输层之间。对所有的分组封装属性标识，其分组结构如表 1 所示，说明如下。

表 1　　属性标识分组结构

分组项	数据长度/bit
版本号	4
次协议	8
长度	8
保留	12
序列号	32
属性标识	64
元数据	128

1) 版本号（4 bit），属性标识的版本。

2) 次协议（8 bit），表示紧跟在属性标识后面的协议类型，如 6（TCP）、17（UDP）或 50（ESP）。

3) 长度（8 bit），其值是以 32 bit（4 B）为单位的整个属性标识的长度（包括头部和元数据）。

4) 保留（12 bit），准备将来对属性标识扩展时使用，目前协议规定这个字段应该被置为 0。

当目的设备 D_2 接收需要移除 IP 分组中的（属性标识，Sign），目的设备 D_2 根据属性标识的起始位置和字段偏移量解析出（属性标识，Sign），并将其从 IP 分组中移除，同时将 IP 头部的协议类型修改为未加属性标识前的上层协议。例如，若（属性标识，Sign）之后的上层协议为 TCP，则将 IP 头部的协议类型修改为 6。

4.3.2 基于属性标识的分组处理模型

为了实现基于属性标识的分组转发和认证，设计了一个基于属性标识的分组处理模型，该模型利用 SDN 的特点设计了基于属性标识的多级流表匹配转发机制，通过流表匹配的方式选择合适的分组进行数据来源验证和实现基于属性标识的匹配转发。如图 5 所示，该模型包括认证交换机和 SDN 控制器 2 个部分。认证交换机主要用于分组的解析、验证和转发，其中包括分组解析模块、数据来源鉴别模块和流表匹配模块。SDN 控制器负责为未匹配的分组下发合适的流规则，对其进行扩展增加了分组解析模块、属性标识失效模块和流规则生成模块，其功能分别为分组解析出属性标识、进行属性标识失效标记和生成匹配的流规则。

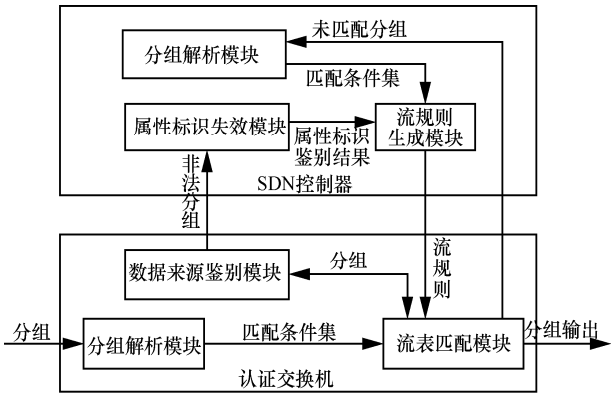


图 5　基于属性标识的匹配转发模型

4.3.3 认证交换机

1) 分组解析模块

认证交换机接收到的分组首先对分组进行解析，利用偏移字段从物理层—数据链路层—网络层—属性标识—传输层的网络层次对分组头进行解析，将分组解析成交换机可识别字段，如 IP 地址、

IP 协议类型、属性标识和 TCP/UDP 端口等。

2) 数据来源鉴别模块

数据来源鉴别模块主要功能为生成设备—端口号映射表和分组合法性验证。

① 设备—端口号映射表

当设备与交换机相连后，交换机与设备通信获取设备地址，将设备地址与路由端口进行一一映射。其主要在鉴别分组合法性时，为认证交换机提供目的设备地址，使认证交换机与目的设备通信获取访问结构 T 的公开参数。

② 分组合法性验证

从流表匹配模块接收到数据流匹配集后，首先获取分组匹配集的目的地址，并通过设备—端口表与目的设备通信获取对应 T 的公开参数，并对数字签名进行验证，将验证通过的分组返回流表匹配模块进行匹配。

3) 流表匹配模块

① 基于属性标识的流表扩展

流表匹配模块将接收匹配条件集与匹配域进行匹配。由于属性标识的加入，需要对原有匹配域进行扩展。根据 OpenFlow 1.3[18]，使用 OXM 架构的 TLV 格式来定义匹配域字段，并将属性标识添加至实验字段 OFPXMC_EXPERIMENTER，在协议的 Flow-Mod 集中增加了属性标识，将带有属性标识的流规则进行匹配。通过这种方式增加了匹配字段，扩展了匹配范围，实现了新的流表匹配规则。其结构如图 6 所示。

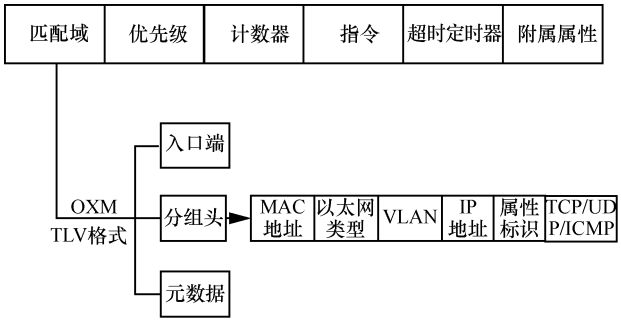


图 6　基于属性标识的 SDN 流表匹配结构

② 端口分类

当数据流进入认证交换机时，有以下 4 种流动方式。流动方式 1，从当前认证交换机到其他位置交换机；流动方式 2，源设备接收后直接转发至目的设备，即源设备和目的设备连接在同一个认证交换机上；流动方式 3，其他认证交换机接收后转发至目的设备；流动方式 4，从源设备接收后转发至其他位置认证交换机。不同流动方式其多级流表处理方式也不同。流动方式 1 和流动方式 4 中，数据流在 SDN 中不需要进行数据合法性验证，直接进行转发；流动方式 2 和流动方式 3 中，数据流要流出 SDN，需要对数据流合法性进行验证。根据上述分析，对认证交换机端口进行分类，如图 7 所示。

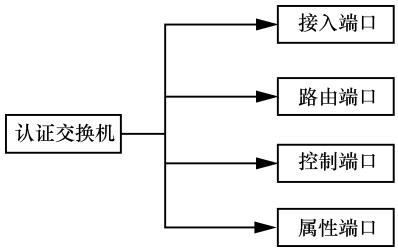


图 7　认证交换机端口分类

认证交换机主要分为接入端口、路由端口、控制端口和属性端口。接入端口与设备相连，路由端口与数据层中其他认证交换机相连，控制端口与控制层控制器相连，属性端口与数据来源鉴别模块相连。不同端口连接的设备不同，故可以通过端口变化和属性标识表示数据流向。流动方式 1 表示为从路由端口进从路由端口出，流动方式 2 表示为从接入端口进从接入端口出，流动方式 3 表示从路由端口入从接入端口出，流动方式 4 表示为从接入端口入从路由端口出。使用 Metadata 字段编码区分数据流向。00 表示待验证的分组，01 表示入端口为接入端口的分组，10 表示接入端口为路由端口的分组，11 表示接入端口为属性端口的分组。初始标记为 00，将解析字段和 Metadata 字段作为匹配条件集发送给流表匹配模块进行匹配查找。

认证交换机通过 OFPT_MULTIPART_REPLY 向控制器发送端口分类信息，使控制器能够对不同流进行划分，根据流向下发对应的流规则。当端口类型发生变化时，认证交换机通过控制端口向控制层控制器发送 OFPT_PORT_MOD 消息，通知端口变更信息。

③ 基于属性标识的多级流表匹配

由于不同流向的数据流处理方式不同，本文使用多级流表[24]的方式进行处理，根据输入端口的不同分配不同的流表，通过流表匹配的方式实现数据转发及合法性验证。

流表匹配模块作为分组处理的主要模块，使用“端口匹配表—属性标识鉴别表—基本转发表”多

级流表转发模式实现基于属性标识的分组转发功能。多级流表分为端口表、属性标识鉴别表、属性标识动作表和基本转发表。端口匹配表作为 0 级流表，用于区分数据来源；属性标识鉴别表作为 1 级流表，用于鉴别属性标识的有效性，其在生命周期内存储从控制器接收的最近一次无效流表，若属性标识匹配则丢弃分组；属性标识动作表作为 2 级流表，通过匹配判断分组是否需要鉴别数据合法性；基本转发表作为 3 级流表，主要用于转发收到的分组，实现分组转发的基本功能。其结构如图 8 所示。

端口匹配表首先与匹配条件集进行匹配，若入端口为接入端口，且 Metadata 字段为 00，则其流表匹配过程如下。

步骤 1　将 Metadata 字段修改为 01，并将匹配集转发至属性标识鉴别表。若匹配或属性标识为空则丢弃分组，若不匹配则转发至属性标识动作表。如图 8 中 a 所示。

步骤 2　匹配属性标识动作表。若匹配则按照指令对数据流进行相应的处理，若直接转发至基本转发表则表示该数据流为接入端口进路由端口出，若转发到数据来源鉴别模块则表示该数据流为接入端口入接入端口出。匹配失败则通过 Packet-In 将数据流首部发送至控制器。如图 8 中 b 所示。

步骤 3　数据来源鉴别模块收到属性标识鉴别表发来的数据流后对数据合法性进行验证，将其返回端口表进行重新匹配，若不匹配则将其上传给控制器的属性标识失效模块。如图 8 中 c 所示。

步骤 4　端口表与返回数据流条件集进行匹配，若匹配端口为属性端口且 Metadata 字段为 11，则直接转发至基本转发表。如图 8 中 d 所示。

步骤 5　基本转发表对条件集进行匹配，按匹配成功顺序进行转发。如图 8 中 e 所示。

若入端口为路由端口，则 Metadata 字段为 10，其流表匹配过程如下。

步骤 1　端口匹配表将匹配集转发至属性标识动作表。若匹配成功，按指令对数据流进行处理，如果直接转发，则表示数据流是属于路由端口入路由端口出；如果将数据流发送至数据来源鉴别模块，则表示数据流属于路由端口入接入端口出，即下一跳为目的设备。若匹配失败，则通过 Packet_in 将数据流首部发送至控制器。

步骤 2　数据来源鉴别模块对数据合法性进行验证，将合法数据返回流表匹配模块进行匹配。

步骤 3　若端口匹配表匹配端口为属性端口且 Metadata 字段为 11，则直接转发至基本转发表。

步骤 4　基本转发表实现数据流匹配转发。

4.3.4　SDN 控制器

SDN 控制器主要任务是给未匹配的分组下发流规则和为非法分组下发失效流表。SDN 控制器在 OpenDaylight 控制器的基础上增加了分组解析功能，能够根据偏移量解析出属性标识；开发了属性标识失效模块标记非法分组；对流规则生成模块进行了扩展，设计了基于属性标识的流规则生成算法，根据匹配域生成带有属性标识的流规则。

1) 标记非法分组

属性标识失效模块从数据来源鉴别模块接收非法分组，获取该分组的源 IP 地址和属性标识，并将其提供给流规则生成模块，生成相应的属性标识失效流表，将失效流表下发至流表匹配模块的属性标识鉴别表，在分组入口禁止该类分组在 SDN 中传送。

2) 基于属性标识的流规则生成

流规则模块根据分组匹配集、属性标识鉴别结果、控制层应用策略和数据层访问策略等条件，生成流规则。流规则分为静态流规则和动态流规则。

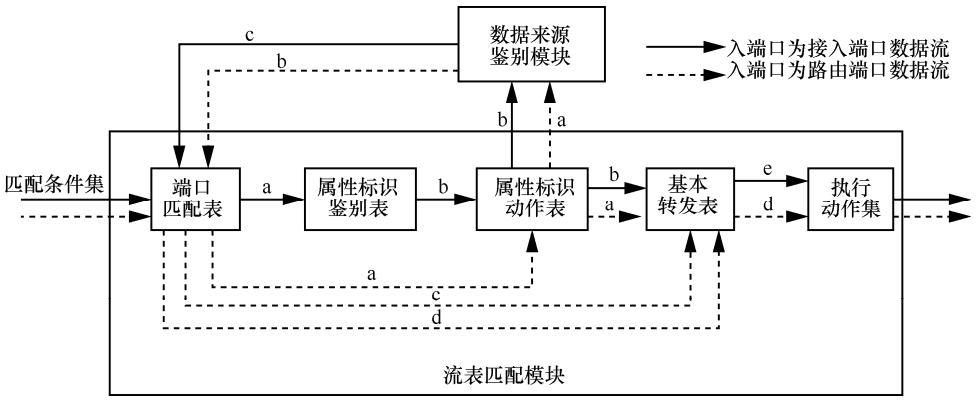


图 8　基于属性标识的多级流表结构

静态流规则即控制机根据认证交换机信息主动下发并预置在认证交换机中的流规则，不会因数据流改变而更新。动态流规则是控制器根据数据层请求被动下发的流规则，根据数据流的改变而更新。

端口匹配表中的流规则为静态流规则，控制器获取全局端口信息后，根据端口信息主动将端口匹配流规则下发给认证交换机端口匹配表。

属性标识鉴别表中流规则为动态流规则，控制器根据属性标识失效模块提供的失效策略生成失效流规则，并将该流规则下发至属性标识鉴别表。

属性标识动作表中流规则为动态流规则，控制器根据数据流的流向、数据流的属性标识和全网拓扑生成相应的流规则，并下发给对应的认证交换机。针对数据流向，设计动作选择算法如算法 1 所示。

算法 1　动作选择算法

输入　认证交换机、未匹配分组

输出　属性表示动作流规则 R

1) if 数据流出端口为路由端口

2) 生成动作为直接转发至基本转发表的流规则 R_2

3) else

4) 生成动作为转发至数据来源鉴别模块的流规则 R

5) 下发流规则 R 到认证交换机

4.3.5　数据分组处理流程

根据以上分析，给出分组处理流程如图 9 所示。认证交换机收到分组后对分组进行解析，并将解析出的匹配集与流表匹配模块进行匹配转发，在匹配过程中，需要将网络出口处分组交由数据来源模块进行鉴别，将未匹配的分组和非法分组交由控制器分别生成新的流规则和失效流规则。

5　基于属性标识的属性签名方案

本文对 Khader[25]的基于属性的群签名方案进行了扩展，使其能够适应 SDN 的分布式情况。通过属性标识组件、属性标识认证中心和认证交换机

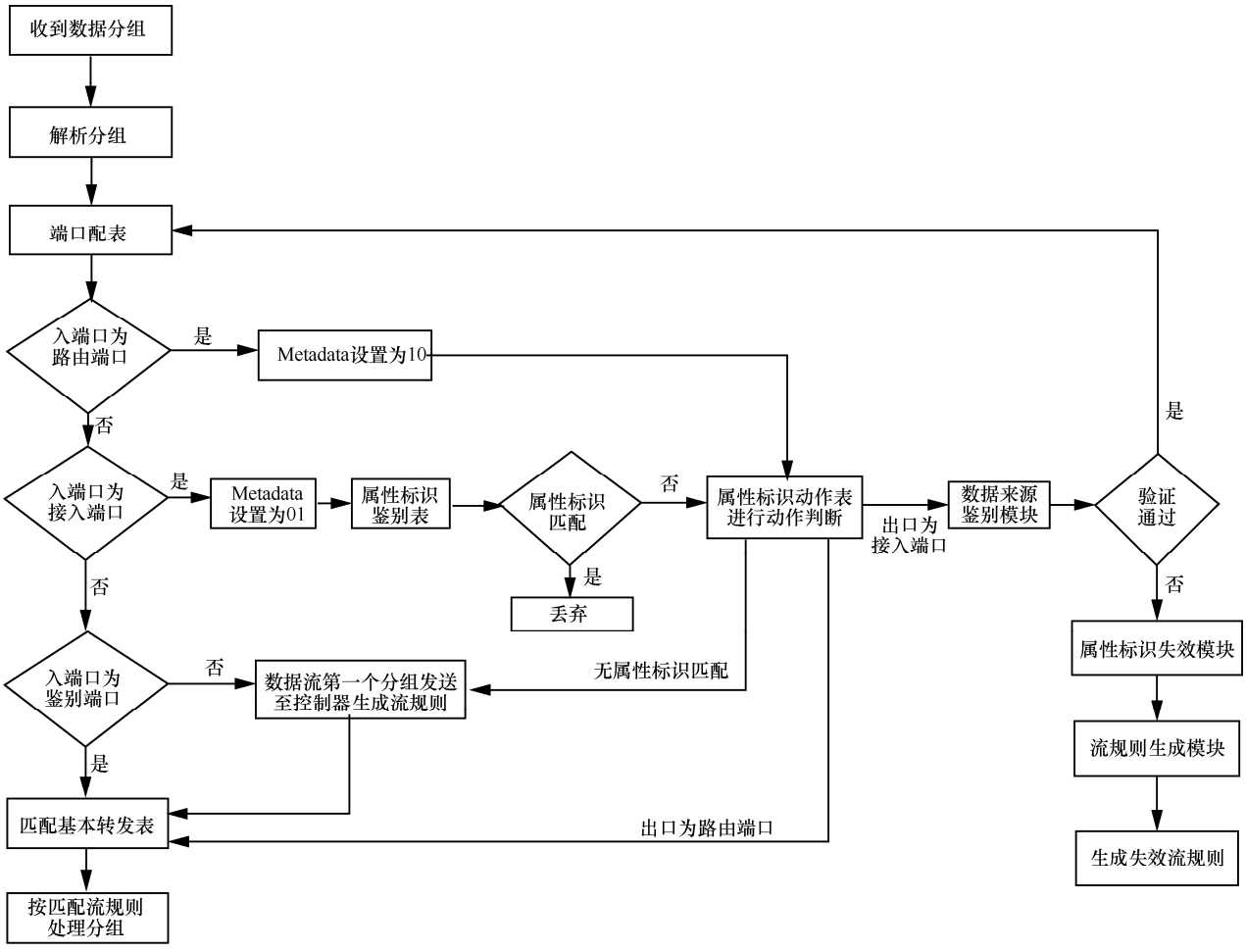


图 9　分组处理流程

实现分组中属性签名的验证和访问控制结构的更新。其过程如图 10 所示。

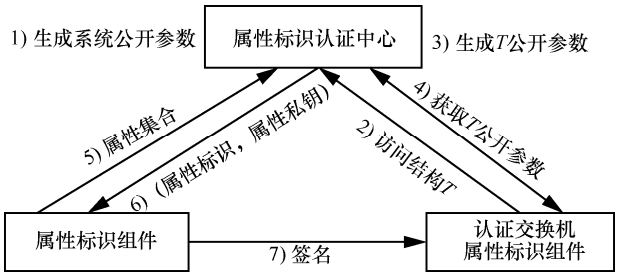


图 10　属性签名模型

1) 属性标识认证中心进行初始化，生成系统公开参数。2) 每个设备的属性标识组件根据需求构造访问控制结构 T，并将其上传给属性标识认证中心。3) 属性标识控制中心使用主私钥与系统公开参数生成 T 的公开参数，将其存储在设备中。4) 属性标识认证中心将公开参数发送给认证交换机。5) 属性标识组件将设备属性集合上传至属性标识认证中心。6) 属性标识认证中心根据设备属性生成属性标识和属性私钥，并将其返回至属性标识组件。因此，不同设备拥有不同的属性标识和属性私钥。7) 当设备间需要通信时，属性标识组件生成带属性标识和签名的分组，并发送至认证交换机，认证交换机获取设备公开参数，并在网络出口对分组进行签名验证。当设备需更新访问结构时，只需上传新的访问结构 T 给属性标识认证中心，获取新的访问结构 T 的公开参数。

5.1　访问结构构造

访问结构 T 为设备收到签名后验证签名所需的属性集，并使用属性树 Γ 来描述，其构造方法基于 Goyal 等[26]的构造方法。在属性树中每个根节点作为一个门限值，并且每个属性作为其叶子节点与其连接。每个门限值表示在其所连的叶子节点中需要满足的条件数，即该根节点下需要的属性数量，其属性树结构如图 11 所示。

利用属性树生成公钥，只有满足属性树中属性要求的用户的数字签名才能通过验证，如图 11 所示，一个 IT 部门的管理员需要进行 read 操作，该用户满足属性树，故其签名可以通过验证，如果同一个 IT 部门的工程师想要进行 read 操作，其不符合属性树的要求，故不能通过验证。

5.2　方案形式化定义

下面给出基于属性的群签名的相关定义。

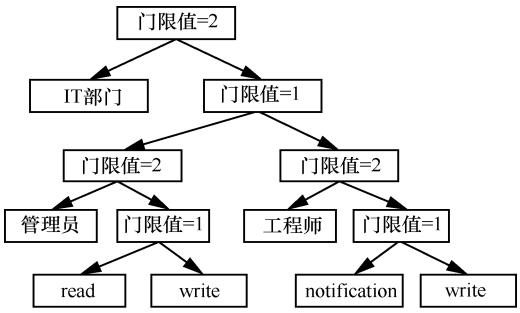


图 11　属性树结构

定义 1　属性树为 Γ，访问控制结构为 T，属性树使用 top—down—left—right 顺序。根节点表示为 (m,n)，m 表示门限值，n 表示其叶子节点的个数；κ 表示属性树的叶子数。则图 10 中的属性树可表示为

$$\Gamma=\{(2,2),\text{IT部门},(1,2),(2,2),(2,2),\text{管理员},(1,2),\text{工程师},(1,2),\text{read},\text{write},\text{notification},\text{write}\}$$

将 Γ 上属性进行 Hash 运算，生成定长字符串。

定义 2　γ_i 表示每个用户所拥有的私钥，μ 表示私钥数即 γ_i 的数量。

定义 3　ζ_i 表示用户用于签名所使用的属性，ζ_i 中的元素满足 γ_i，即 $\zeta_i\subseteq\gamma_i$，例如 $\Gamma=\{(1,2),$ 管理员,工程师$\}$，员工 i 使用 $\zeta_i=\{$工程师$\}$ 可以通过验证，τ 表示 ζ_i 的元个数。

该算法包括 Setup、KeyGen、Sign 和 Verify 共 4 个部分。

Setup 属性标识认证中心选择一个双线性对 $e:G_1\times G_2\to G_T$，其中，G_1、G_2 和 G_T 是 p 阶乘法循环群，p 是一个大素数。g_2 是 G_2 的生成元，存在同态映射 $g_1\leftarrow\psi_2$，其中 ψ_2 为属性集合。系统选择一个公开的 Hash 函数 $H:\{0,1\}^*\to Z_P^*$，属性散列结果集合为 Z_P^*；选择 $h\in G_1$ 和随机的 ξ_1 和 ξ_2（$\xi_1,\xi_2\in Z_P^*$），设 $u,v\in g_1$ 且 $u^{\xi_1}=v^{\xi_2}=h$，随机选择 $\omega\in Z_P^*$，并计算 $W=g_2$。定义属性空间 $U=\{1,2,\cdots,n\}$，每个 $j\in U$ 选择属性集合 $t_f\in Z_p^*$，然后，计算出公开参数 PK 和系统秘密参数 MK。

$$\text{PK}=\{G_1,G_2,G_T,\text{g}_1,g_2,e,H,h,u,v,W\}$$

$$\text{MK}=<\{t_i\}_{i\in U},\omega,\xi_1,\xi_2>\text{KeyGen}$$

该过程包括为用户生成属性私钥和为访问控制结构 T 生成公开参数两部分。系统通过 γ 为用户 i $(1\leqslant i\leqslant n)$ 生成一个基私钥 $\text{gsk}[i]_{\text{base}}=(A_i,x_i)$，并

且其是一个 SDH（strong Diffie-Hellman）对，即 $A^i = g_1^{\frac{1}{\omega+x_i}} \in G_1$。

1) $\text{KeyGen}_{\text{public}}(\Gamma)$。根据属性树 Γ 的结构，首先为 Γ 中每个非叶子节点选择一个多项式 q_x，对于根节点使用自上而下的构造方式。首先，树中节点为 x，其多项式 q_x 的次数 d_x 小于其阈值 k_x，即 $d_x = k_x - 1$。作为起始项的根节点 $q_r(0) = \omega$。然后，随机选取其他节点的多项式，其他节点满足 $q_n(0) = q_{\text{parent}}(\text{index}(x))$。随机选取多项式 q_n 进行递归构造属性树多项式。最后，通过以下计算式

$$D_x = g_2^{\frac{q_x(0)}{t_j}}$$

$$h_n = h^{t_i}$$

$$i = \text{att}(x)$$

获取属性树 Γ 的公开参数 $\text{TPK} = <\{D_x, h_x\}_{x\in\gamma_\Gamma}>$，并将 TPK 和系统公开参数 PK 发送给认证交换机的分组来源鉴别模块。

2) $\text{KeyGen}_{\text{private}}(\text{gsk}[i]_{\text{base}}, \gamma_i)$。为拥有属性 γ_i 的用户 i 颁发私钥。对于属性 $j \in \gamma_i$，计算 $\Gamma_{i,j} = A^{t_j}$ 获得用户私钥为 $\text{SK} = <A_i, x_i, \{T_{i,j}\}_{j\in\gamma_i}> \text{Sign}$。当系统输入属性集合 $\gamma_i \subseteq U$，$j \subseteq \gamma_i$，属性树的公开参数和消息 m 后，进行以下步骤。

步骤 1 选择属性签名 $\zeta \subseteq \gamma_i$ 和随机数 α、β，$\text{rnd} \in Z_p^*$。

步骤 2 计算 A_i 和 T_{ij} 的线性加密，其中 $j \in \zeta$，计算式为

$$C_1 = u^a$$

$$C_2 = v^\beta$$

$$C_3 = A_i h^{\alpha+\beta}$$

$$\text{CT}_j = (T_{i,j} h_j^{\alpha+\beta})^{\text{rnd}}$$

步骤 3 计算 $\delta_1 = x_i\alpha$，$\delta_2 = x_i\beta$，然后选择随机数 r_α、r_β、r_x、r_{δ_1} 和 r_{δ_2}，通过随机数计算 $R_1 = u^{\gamma_\partial}$，$R_2 = v^{\gamma_\beta}$，$R_3 = \hat{e}(C_3, g_2)^{\gamma_x} \hat{e}(h,\omega)^{-\gamma_\alpha-\gamma_\beta} \cdot \hat{e}(h, g_2)^{-\gamma_{\delta_1}-\gamma_{\delta_2}}$，$R_4 = C_1^{\gamma_x} u^{-\gamma_{\delta_1}}$ 和 $R_5 = C_2^{\gamma_x} v^{-\gamma_{\delta_2}}$。

步骤 4 $c = H(M, C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5) \in Z_p^*$。

步骤 5 构造值 $s_\alpha = (\gamma_\alpha + c\alpha)$，$s_\beta = (\gamma_\beta + c\beta)$，$s_x = (\gamma_x + cx)$，$s_{\delta_1} = (\gamma_{\delta_1} + c\delta_1)$ 和 $s_{\delta_2} = (\gamma_{\delta_2} + c\delta_2)$。

步骤 6 使 $\eta = \omega^{\text{rnd}}$，计算出数字签名 $\sigma = (m, C_1, C_2, C_3, c, \{\text{CT}_i\}_{i\in\zeta}, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}, \eta)$，最后将签名及属性标识 (σ, AID) 一同发给验证者，其中 AID 是属性集 ζ 的 Hash 运算。

Verify 收到签名后对签名进行验证。首先，验证签名人是否满足访问结构 T，定义一个递归算法 Verifynode，对于叶子节点 x，进行如下计算

$$\text{Verifynode}(x) = \begin{cases} \hat{e}(\text{CT}_j, D_j), j = \text{att}(x), j \in \zeta \\ \text{return } \perp，\text{其他} \end{cases}$$

结果为 $\hat{e}(\text{CT}_j, D_j) = \hat{e}(A_i h^{\alpha+\beta}, g^{\text{rnd}}{}_2)^{q_j(0)}$。

假如 x 节点不是叶子节点，执行以下步骤。将所有 x 子节点 z 值 $\text{Verifynode}(z)$ 保存在函数 F_z 中，使用拉格朗日插值法递归求出根节点的 F_x 值。在 $j \in \{\text{index}(z) : z \in s_x - \text{index}(z)\}$ 条件下，计算 $\Delta s_{x,\text{index}(z)} = \prod -\frac{j}{\text{index}(z) - j}$，并且进行如下计算获取父节点值。

$$F_x = \prod_{z\in s_x} F_z^{\Delta s_{x,\text{index}(z)}}$$

$$F_x = \prod_{z\in s_x} (\hat{e}(A_i h^{\alpha+\beta}, g_2^{\text{rnd}})^{q_z(0)})^{\Delta s_{x,\text{index}(z)}}$$

$$F_x = \prod_{z\in s_x} (\hat{e}(A_i h^{\alpha+\beta}, g_2^{\text{rnd}})^{q_{\text{parent}(z)}(\text{index}(z))})^{\Delta s_{x,\text{index}(z)}}$$

$$F_x = \hat{e}(A_i h^{\alpha+\beta}, g_2^{\text{rnd}})^{q_x(0)}$$

递归求出根节点 F_x 的值，验证 $F_x = \hat{e}(C_3, \eta)$，如果成立则表示签名满足属性树 Γ，则进行如下计算过程，否则拒绝签名。

$$\overline{R}_1 = u^{s_\alpha} C_1^{-\kappa},$$

$$\overline{R}_2 = v^{s_\beta} C_2^{-\kappa},$$

$$\overline{R}_3 = \hat{e}(C_3, g_2)^{s_x}\ \hat{e}(h, W)^{-s_\alpha - s_\beta} \hat{e}(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \left(\frac{\hat{e}(C_3, W)}{\hat{e}(g_1, g_2)}\right)^{\kappa},$$

$$\overline{R}_4 = C_1^{s_x} u^{-s_{\delta_1}},$$

$$\overline{R}_5 = C_2^{s_x} v^{-s_{\delta_2}}$$

若满足 $\kappa = H(m, C_1, C_2, C_3, \overline{R}_1, \overline{R}_2, \overline{R}_3, \overline{R}_4, \overline{R}_5)$，签名通过验证。

5.3 方案安全性分析与证明

由于匿名性和不可伪造性是安全群签名方案

基本安全要求[30]，因此在本节对方案的正确性、匿名性和不可伪造性进行证明。

1) 方案正确性证明

由于 $c=H(M,C_1,C_2,C_3,\overline{R}_1,\overline{R}_2,\overline{R}_3,\overline{R}_4,\overline{R}_5)$，当 $\overline{R}_1=R_1$，$\overline{R}_2=R_2$，$\overline{R}_3=R_3$，$\overline{R}_4=R_4$，$\overline{R}_5=R_5$ 时签名验证通过。

$$\overline{R}_1=u^{s_\alpha}C_1^{-c}=u^{\gamma_\alpha}(u^\alpha)^{-c}=u^{\gamma_\alpha}=R_1$$

$$\overline{R}_2=v^{\gamma\beta+c\beta}(v^\beta)^{-c}=v^{\gamma\beta}=R_2$$

$$\overline{R}_3=R_3$$

$$\overline{R}_4=C_1^{s_x}u^{-s_{\delta_1}}=u^{\alpha(\gamma_x+cx)}u^{(-\gamma\delta_1-c\delta_1)}=C_1^{\gamma x}u^{-\gamma\delta_1}=R_4$$

$$\overline{R}_5=C_2^{s_x}v^{-s\delta_2}=v^{\beta(\gamma_x+cx)}v^{(-\gamma\delta_2-c\delta_2)}=C_2^{\gamma_x}v^{-\gamma\delta_2}=R_5$$

上述等式成立的理由为

$$\hat{e}(C_3,g_2)^{s_x}\hat{e}(h,\omega)^{-s_\alpha-s_\beta}\hat{e}(h,g_2)^{-s_{\delta_1}-s_{\delta_2}}=$$

$$\hat{e}(C_3h^{-\alpha-\beta},\omega g_2^x)\hat{e}(C_3,\omega)^{-c}(R_3)=$$

$$\left(\frac{\hat{e}(A,\omega g_2^x)}{\hat{e}(C_3,\omega)}\right)^c R_3=\left(\frac{\hat{e}(g_1,g_2)}{\hat{e}(C_3,\omega)}\right)^c R_3$$

2) 匿名性证明

假如线性加密能够抵抗不可区分性选择明文攻击（IND-CPA, indistinguishability under chosen-plaintext attack），那么该方案具有匿名性。下面用反证法进行证明。

证明　假设攻击者 A 能打破 SSA 安全，则攻击者 E 能打破 IND-CPA 安全，攻击者 A 和 E 共同扮演一个挑战者，当 A 发起挑战时，他发送 i_0、i_1、消息 M 和集合 ζ 给 E，E 运行 Setup 步骤，获取 A 的基私钥 A_{i_0}、A_{i_1}，通过这 2 个值对线性加密方案发起 IND-CPA 挑战。E 将会得到这 2 个值中一个值 A_{i_b} 的一份密文。其密文为 $\overline{C}=\langle C_1,C_2,C_3\rangle$，其中 $C_1=u^\alpha$，$C_2=v^\beta$，$C_3=A_{i_b}h^{\alpha+\beta}$，E 进行计算 $\mathrm{CT}_j=C_3^{\mathrm{rnd}.t_j}$、$c$、$\eta$、$s_\alpha$、$s_\beta$、$s_x$、$s_{\delta_1}$ 和 s_{δ_2}。E 给 A 发送 i_b 的数字签名 $\sigma_b=\langle C_1,C_2,C_3,c,\mathrm{CT}_1,\cdots,\mathrm{CT}_\mu,s_\alpha,s_\beta,s_{\delta_1},s_{\delta_2},\eta\rangle$，但是攻击者 E 自己不知道随机值 b 的取值。假如 A 能够打破该方案的匿名性，则 E 将从 A 处获得正确的 b，通过 b 获知 A_{i_0} 和 A_{i_1} 是否被加密。E 将打破线性加密的 IND-CPA 安全，与条件相矛盾，所以该方案具有匿名性。证毕。

3) 不可伪造性证明

针对该方案主要攻击为替换用户属性私钥攻击，即伪造属性签名 $\sigma=(\sigma_1,\sigma_2)$，其中 σ_1 和 σ_2 分别由用户主私钥和属性私钥构成，分别替换用户的主私钥和属性私钥。本方案在验证时通过 $e(A^{-1},D)=g(g_1,W)$ 将 σ_1 和 σ_2 进行关联，防止主私钥的攻击，验证时加入系统主私钥相关的 W，使攻击者无法替换属性认证中心生成的属性私钥，可以防止属性私钥攻击。根据分析，构造攻击者—挑战者游戏模型来证明方案的不可伪造性，将不可伪造性规约于 SDH 困难问题，通过反证法进行证明，详细证明如附录 A 所示。

6　实验及评估

本文已实现并部署了一个基于属性标识的 ACFlow 原型系统。该系统基于 OpenDaylight 进行二次开发，对 OpenDaylight 控制器和 OVS 交换机进行功能扩展以完成功能部署，并利用 sFlow 监控网络流量；最后通过双线性加密库和 C++代码实现基于属性的群签名方案。接下来将从功能有效性、耗时、性能这三方面对其进行评估。

6.1　实验环境

使用 10 台主机作为实验环境，其配置如表 2 所示。

表 2　　**实验环境配置**

主机功能	数量	配置
控制器	1	CPU i7-8400，内存 16 GB，网卡 4 个
认证交换机	3	CPU i7-8400，内存 16 GB，网卡 6 个
终端	6	CPU i5-8400，内存 8 GB，网卡 2 个

6.2　功能有效性测试

首先，对第 3 节的 2 个攻击场景进行测试，表明通过该系统的保护恶意源设备 A 不能对目的设备 C 实施扫描攻击。

1) 如图 2 所示，恶意源设备 A 通过添加新的匹配流表绕过防火墙直接访问目的设备 C，造成防火墙失效，实现对目的设备的直接攻击。而在基于属性标识的 SDN 中，首先认证交换机会对源设备进行数字签名认证，决定该设备的分组是否合法，由于恶意源设备 A 不满足目的设备 C 的访问结构，其发送数据无法到达目的设备 C。

2) 如图 3 所示，恶意源设备 A 不直接访问目的设备 C，而是通过添加 2 条流规则，使合法源设备 B 访问目的设备 C，然后修改数据转发的目的地址，将目的设备 C 的数据分组转发给恶意源设备 A，

间接实现对目的设备的扫描攻击，造成防火墙的失效。但在 ACFlow 系统中，会对目的设备的设备密码属性进行验证，验证其是否符合目的设备的访问结构，对合法但不符合访问规则的源设备进行屏蔽，实现对目的设备的保护，如源设备 B 对目的设备 C 访问的数据分组是合法的故可以通过，但是目的设备 C 不能满足到设备 A 的访问结构，故目的设备 C 的数据分组在到达 $OpenvSwitch_1$ 时被拦截。

根据以上 2 种情况，对基于属性标识的 SDN 进行功能评估：1) 该系统能否鉴别属性标识的有效性；2) 该系统能否实现基于属性标识的访问控制。根据要求，其实验网络拓扑如图 12 所示。

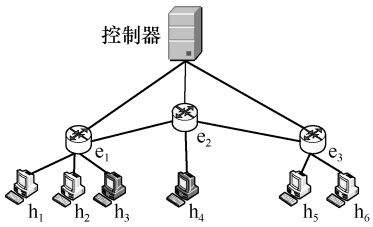


图 12 实验网络拓扑

1) 主机 h_1 携带有效的属性标识和数据签名，h_2 携带不符合 h_6 访问结构的属性标识和数据签名，h_3 为一般分组不携带任何属性标识和数据签名，以上 3 台源设备以 50 package/s 的速率向 h_6 持续发送数据，控制器为其下发路径为 $h_1/h_2/h_3 \rightarrow e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow h_6$，使用 SFlow 对 e_1、e_2 和 e_3 的流量进行监控，重复 10 次，每次 12 s，结果取均值。其结果分别如图 13~图 15 所示。横坐标表示时间，其单位为 s；纵坐标表示数据分组速率，其单位为 package/s，正向代表流入数据分组，即认证交换器接收 h_1、h_2 和 h_3 的数据分组，负向代表流出的数据分组，即向 h_6 转发的数据分组。

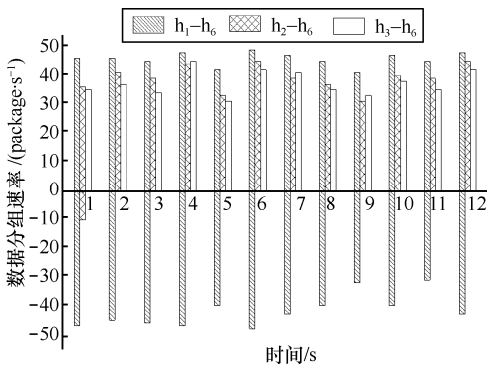


图 13 认证交换机 e_1 流量统计

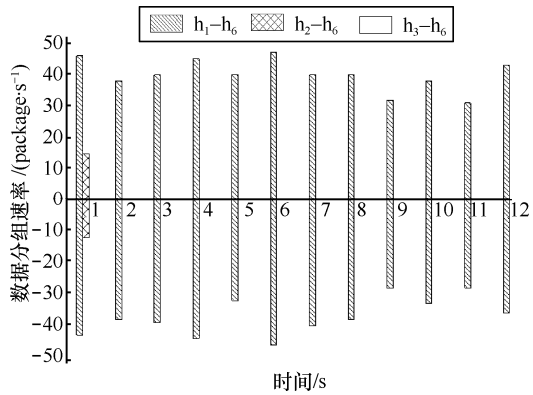


图 14 认证交换机 e_2 流量统计

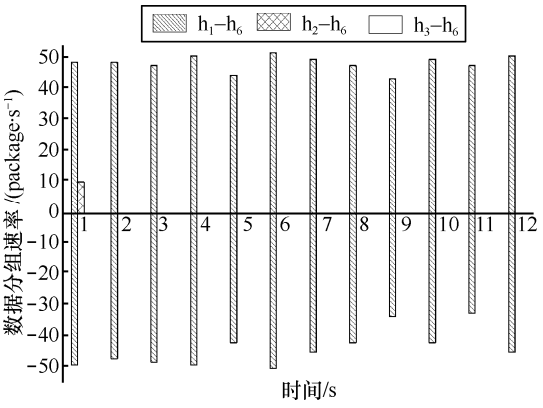


图 15 认证交换机 e_3 流量统计

从图 13 中可以看出，认证交换机 e_1 接收了 h_1、h_2 和 h_3 的数据分组，由于只有 h_1 和 h_2 携带属性标识，因此 e_1 只对 h_1 和 h_2 的数据分组进行转发，h3 数据分组直接丢弃，但 h_2 在第一秒进行了少量转发，随后进行丢弃。

从图 14 和图 15 可知，h_3 的分组已丢弃，图中 h_3–h_6 值为 0，h_2 分组在第 1 s 内进行转发随后被丢弃，h_1 的分组仍在转发。随后查看认证交换机发现，e_2 和 e_3 没有添加新的流规则，e_1 生成一条无效流规则将 h_2 的数据分组丢弃。

从以上实验结果可知，当数据分组进入认证交换机后，首先认证交换机通过属性标识鉴别表将不符合规格即属性标识为空的数据分组丢弃，并对符合转发条件的数据分组进行转发，在目的端对数据分组合法性进行验证，合法数据分组进行转发；对不合法的数据分组，控制器会在接收端下发无效流表对其随后的数据分组进行丢弃。因此，该网络可以将失效的属性标识进行丢弃，并且可以拒绝未携带属性标识的非法数据分组进入网络。

2) h_1 携带符合 h_6 访问结构的属性标识和签名，并以每秒 50 个数据分组的速率向 e_1 发送数据分组，

在前 12 s 主机 h_1 数据分组以路径 $h_1 \rightarrow e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow h_6$ 发送给 h_6，之后 12 s 通过控制器将 h1 的路径修改为 $h_1 \rightarrow e_1 \rightarrow e_2 \rightarrow h_4$，将发往 h_6 的数据分组发送给 h_4。并使用 SFlow 对 e_2 和 e_3 进行流量监控，重复 10 次，每次 12 s，结果取平均值，分别如图 16 和图 17 所示。

从图 16 和图 17 中可知，对于携带相同的属性标识和签名的数据分组，其只能访问符合访问结构的目的设备，当访问其他设备时会被认证交换机判定非法，将其进行丢弃。因此，可以实现基于属性标识的细粒度访问控制。

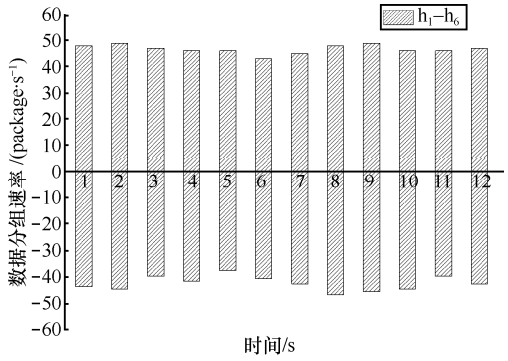


图 16　认证交换机 e_3 流量统计

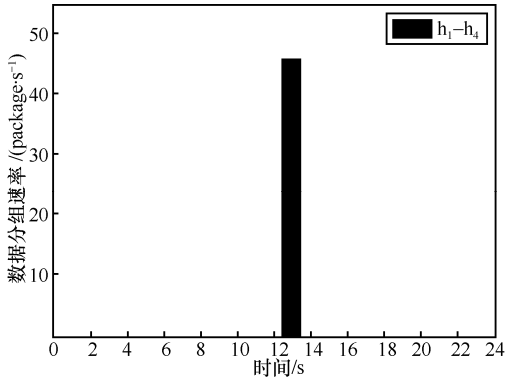


图 17　认证交换机 e_2 流量统计

6.3　方案对比

设计 ACFlow 主要实现数据流的细粒度控制，和实现数据流的有效控制。但是在实际应用中，随着网络规模的增加，用户所需的角色越来越多，其访问控制管理难度也随之增加。

将本文方案与常用的 6 种数据流认证方案进行对比，如表 3 所示，包括 SE-Floodlight[7]、RoseMary[8]、FRESCO[9]、FortNOX[4]、PERM-GUARD[33]和 PermOF[10]，以上方案通过认证方式阻止用户的非法访问。

表 3　　**认证方案对比**

方案	签名方式	权限管理	身份认证粒度
SE-Floodlight	基于角色	基于角色	3 个角色
RoseMary	基于角色	基于角色	3 个角色
FRESCO	基于角色	基于角色	3 个角色
FortNOX	基于角色	基于角色	3 个角色
PERM-GUARD	基于身份	身份认证，访问列表	16 个权限
PermOF	未使用	访问控制列表和 PKI	18 个权限
ACFlow	基于属性	属性认证	自定义访问结构数量

在 SDN 中每产生一条新的流规则，代表一次数据流的改变，需要对用户进行相应身份认证。流规则的数量与身份认证数量呈正相关，其反映的是数据流管理粒度。因此，将本文方案管理粒度与其他方案管理粒度进行对比，结果如图 18 所示。

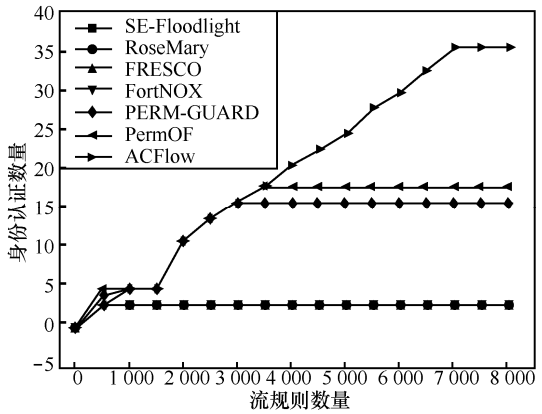


图 18　身份认证粒度对比

由于 SE-Floodlight、RoseMary、FRESCO、FortNOX 提供身份验证只有 3 个角色，因此最先出现身份认证数量瓶颈。PERM-GUARD 和 PermOF 由于使用访问列表，能够提供更细粒度的数据流管理，但是限于访问列表的大小，其在更大数量的流规则下出现了身份认证数量不足的情况。由于本文方案采用属性签名，其根据需求定义访问结构，只有满足其访问结构的签名才能验证成功，该签名算法实现访问控制不需要存储访问列表，因此不会受到传统访问控制中访问列表的限制，能够实现更好的细粒度管理。但是，其最大访问数量会受到 SDN 交换机处理流表能力的限制，当多个数据流通过一个交换机时，会造成流表项过大产生拥塞，所以当流表数量增加到一定程度后 ACFlow 身份认证数量不增加。

6.4　性能分析

本节对基于属性标识的 SDN 性能进行评估，主要考虑以下几个问题：问题 1，基于属性标识的 SDN 分组处理能力；问题 2，控制器 CPU 利用率；问题 3，主机 CPU 利用率。

针对问题 1，基于属性标识 SDN 分组处理能力主要体现在分组签名能力和分组验证能力上，即生成一组带有属性标识和签名的 IP 分组所需的时间和对每组 IP 分组签名进行验证消耗的时间。这两项指标决定网络分组的处理能力。但由于主机发送分组长度不同和不同时间下主机流量不同，故不同时间和不同分组长度的测量结果不同，为了测量最大分组签名和验证能力，需要注意以下两点：1) 尽可能增加签名验证在分组产生和转发时间中所占的比例；2) 在网络最活跃时进行测量，保证测出签名最快速率。对于 1) 通过在主机上持续发送定长为 64 B 的短分组实现。对于 2)需要判断何时为最活跃时间，分别参照斯坦福 300 名用户[27]和 22 000 名用户[28]SDN 实验以及伯克利实验室[27]在超过 8 000 名用户 SDN 中统计数据，表明每个活跃的设备在其运行的 9～25 min 时设备转发分组量最大。因此，首先在主机运行的 9～25 min 内统计了 200 组分组（每个分组定长为 64 B，每组 20 MB），每组分组生成时间如图 19 所示，其中横坐标为分组编号，纵坐标为分组生成所需要的时间。从图 19 中可知，生成新的带属性标识分组的平均时间为 25.76 ms，即每秒可发送 38 组不同属性标识和签名的数据分组，速率约为 760 MB/s。

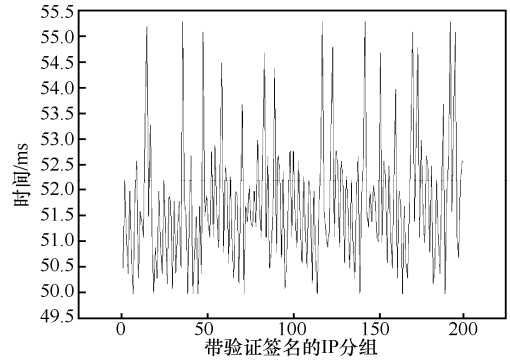


图 19　生成带属性标识和签名的 IP 分组时间

关于认证交换机验证分组签名能力，通过对 200 个带不同属性标识和签名的分组（每个分组 64 B，每组 20 MB）进行验证，得出验证每个分组所需要的时间如图 20 所示。

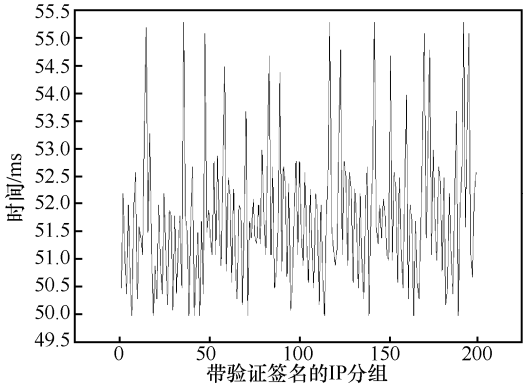


图 20　验证带属性标识和签名 IP 分组时间

通过图 20 可知，认证交换机对带有签名的分组组验证的平均时间为 51.77 ms，表示认证交换机每秒可以处理 386 MB。据文献[7,27-29,33]，日常使用中交换机需要处理流量如表 4 所示。根据表 4 可以得出，ACFlow 系统分组处理能力能够满足网络基本需求。

表 4　不同的网络中交换机平均转发数据分组

方案	网络规模	平均每秒转发数据分组数/(MB·s^{-1})
文献[7]	4 个主机	196
文献[27]	超过 300 名用户	246
文献[28]	22 000 名用户	497
文献[29]	8 000 名用户	356
文献[33]	100 个主机	320

随着认证交换机收到数据的增加，其认证的时间也会相应地增加，而验证效率会否随着数据量增加而降低成为数据吞吐量的瓶颈。为此对不同数据量下认证交换机所需认证时间进行统计，结果如图 21 所示，横坐标代表分组数，其中每组 20 MB。根据结果可知，认证交换机验证分组数量和时间呈线性关系，验证分组的速率不随分组增加而改变。

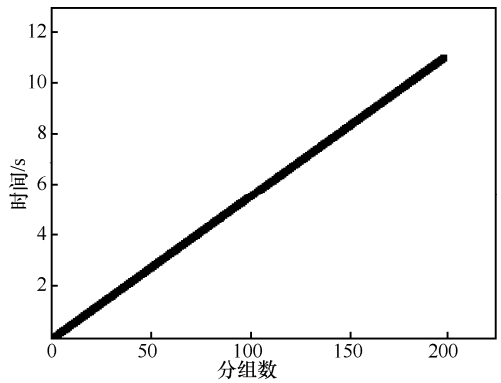


图 21　不同数据量下认证交换机所需认证时间

针对问题 2，使用原生 OpenDaylight 控制器和经过二次开发的 OpenDaylight 控制器进行 CPU 利用率对比。在主机上不同位置持续发送合法的数据分组，统计在不同 Packet_in 速率下控制器 CPU 利用率，结果如图 22 所示。

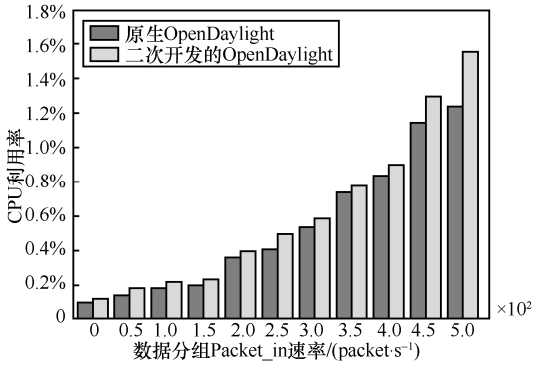


图 22　控制器 CPU 利用率对比

由图 22 可知，在相同 Packet_in 速率下 ACFlow 系统中的控制器 CPU 利用率略高于原生 OpenDaylight 控制器，这表明在实现安全功能的前提下，ACFlow 系统没有过多增加控制器负载。

针对问题 3，在 h_1、h_2 上使用 hping3 向网络注入数据分组，其速率分别为 0、50 Mbit/s、100 Mbit/s、150 Mbit/s、200 Mbit/s、250 Mbit/s、300 Mbit/s、350 Mbit/s、400 Mbit/s、450 Mbit/s、500 Mbit/s，在不同场景下分别测量带属性标识组件主机和标准主机的 CPU 利用率。在不同的速率下重复 10 次。每次取其平均值，其结果如图 23 所示。

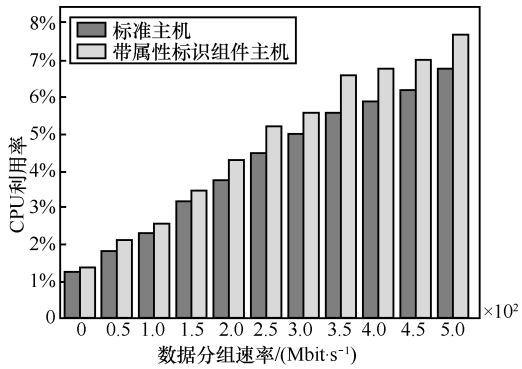


图 23　主机 CPU 利用率对比

从图 23 可知，带属性标识组件的主机 CPU 占用率略高于标准主机，二者都随着流量的增大而增大。随着数据流量的增大其所需封装的数据分组增加，CPU 占用率差值逐渐增大。因为带属性标识组件的主机要在发送数据分组之前生成签名并对签名和属性标识进行封装，虽然生成签名的计算开销大，但其只与设备属性有关，与数据分组无关，只需进行一次运算，而封装签名和属性标识的计算开销很小。因此在分组数目可控的情况下，随着速率增加二者 CPU 占用率差别较小，最大相差 0.8%。

6.5　网络可用性分析

通过与不加入属性标识和签名的 SDN 中分组丢失率和时延进行对比，对网络可用性进行分析。首先在设备上运行 WireShark，以不同的速率、不同端口发送分组长为 1 514 B 的数据分组，并统计其分组丢失率如图 24 所示。

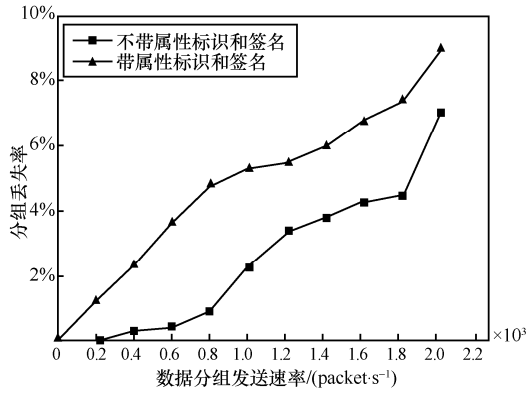


图 24　网络分组丢失率

从图 24 可以看出，随着分组发送率增大，网络的分组丢失率增大，由于携带属性标识和签名的数据分组在交换机中需要进行验证和多级流表匹配等操作，因此其分组丢失率大于不带属性标识和签名的网络，结果表明该架构额外带来的网络分组丢失率为 2.56%。

然后，统计 50 次分组 ACK 返回时延，重复 10 次求其平均值。在 ACFlow 架构下，每次分组 ACK 返回时延记为 T_{ACFlow}，而不加入属性标识和签名的系统其返回时延记为 $T_{\mathrm{N\text{-}ACFlow}}$，其测量数据如图 25 所示。

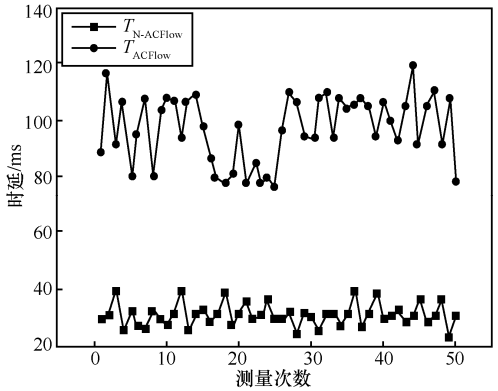


图 25　网络时延对比

从图 25 可知，未添加属性标识和签名的网络延迟的平均值为 31.48 ms，而 ACFlow 架构的网络延迟平均值为 103.2 ms，其平均时延增加了 71.72 ms。通过上面分析可知，生成属性标识和签名并对签名进行验证的时间占整个时延的 65.4%，所以签名的算法复杂度直接决定了网络时延，但是仍在可行通信时延内。

7 结束语

针对 SDN 数据来源验证和细粒度的访问等问题，通过软件定义网络和密码技术的融合，提出了基于属性标识的 ACFlow 架构，该架构利用属性标识的身份属性、认证属性和标识属性，在网络进出口处对属性标识进行验证，实现了数据流合法认证。通过密码标识流表匹配机制，实现基于密码标识的流表匹配，能够对网络进行灵活的动态管理，实现网络访问的细粒度管理，实现基于人、事、物的多维划分，并且在安全功能上实现了去中心化，将安全功能分布在数据层中，降低了对控制器影响。

在网络的功能性和可用性上进行了验证，验证结果表明该架构在实现网络安全功能的基础上保证了网络的可用性。相似认证方案的对比表明，ACFlow 在认证粒度方面具有较大优势。虽然 ACFlow 能够实现数据流的认证管理，但其仍面临 SDN 北向开放接口所带来的安全威胁，未来将从北向接口流规则认证方面和数据流验证两方面做出更多努力，以此提高 SDN 整体安全性。

附录 A

匿名性证明 定义一个攻击者 A 与挑战者 C 的游戏模型。

初始化 首先 A 选定要挑战的访问结构 T，伪造满足 T 的用户签名。

参数生成 C 运行方案中的 Setup 算法，选定 (G_1,G_2) 以及 (g_1,g_2)。并随机选择与属性相对的 $t_1,\cdots,t_n$ 和其私钥 s。随机选择用户的私钥对 (A_i,x_i)，在这些私钥对中设定 A_i 对应的 x_i 不可见，其他的私钥对则是一个有效的 SDH 对，随后 C 为 T 构造相同的公开参数并发送给敌手 A。敌手的参数为 $\mathrm{PK}=\{g_1,g_2,h,u,D_{\mathrm{leaf}_1},\cdots,\ D_{\mathrm{leaf}_n},h_1,\cdots,h_n\}$。

散列查询 A 可以对 $H_3(C_1,C_2,C_3,\ R_1,R_2,R_3,R_4,R_5)$ 进行查询，C 将随机返回一个 G_1 元素给 A，并且保留结果防止相同的查询。

私钥查询 A 要求查询给定 i 的属性，结合 γ 的私钥进行查询，如果 x_i 不为空，C 返回 (A_i,x_i) 给 A；否则，C 宣布游戏失败。

签名查询 A 要求查询用户 i 的属性集合 ζ 在消息 M 上的签名，有 ζ 满足访问结构 T。如果 x_i 不为空，C 则计算 $T_{i,j}=A_{i,j}$，并按算法得到签名 σ，并将其发送给 A。过 x_i 为空，则 C 随机选择 $\gamma_1,\gamma_2,\gamma_A,\gamma_B\in Z_p^*$，然后进行以下计算

$D_{B,j}=(A_ih^{r_1})^{r_2}$，$\sigma_2=g_2^{\gamma_B}$，$V'_\omega=g_2^{\gamma_2,\gamma_A}$，$C_1=u^{r_1}$，$C_2=v^{\gamma_2}$，$\sigma_1=D_{B,j}H_2(\mathrm{IDA},m,m_\omega,g_2^{\gamma_B})^{\gamma_B}$，$C_3=A_ih^{\gamma_1+\gamma_2}$，$\mathrm{CT}_j=(T_{i,j}h_j^{\gamma_1+\gamma_2})^{\mathrm{rnd}}$。

随机选择 $\kappa,s_1,s_2,s_3,s_4\in Z_p^*$。计算 $R_1=u^{s_1}C_1^{-\kappa}$，$R_2=u^{s_2}C_2^{-\kappa}$，$R_3=\hat{e}(C_2,g_2)^{s_4}\hat{e}(h,W)^{-s_1-s_2}\ \hat{e}(h,g_2)^{-s_3-s_4}\cdot\left(\dfrac{\hat{e}(C_3,W)}{\hat{e}(g_1,g_2)}\right)^{\kappa}$，$R_4=C_1^{s_1}u^{-s_3}$，$R_5=C_2^{s_1}v^{-s_4}$。为了保持一致性，保存 κ 到 $H(C_1,C_2,C_3,R_1,R_2,R_3,R_4,R_5)$ 中，同时计算出 W。

将签名 σ 发送给 A。对于这个签名 A 可以通过验证，认为是有效签名。

现在通过 A 的伪造构造一个新的 SDH 对，这需要用到分叉引理证明。假设敌手伪造签名为，$\sigma=(\kappa,\sigma_1,\sigma_2,\sigma_3,\sigma_4)$，具体定义如下。$\sigma_1=(C_1,C_2,C_3,R_1,R_2,R_3,R_4,R_5)$，其中 κ 来自 σ_1 的散列值。$\sigma_2=(s_1,s_2,s_3)$ 用于计算散列缺失的输入，$\sigma_3=(\{\mathrm{CT}_j\}_{j\in\zeta},W)$，这个取决于每次签名的属性集合，$\sigma_4$ 为签名的其他部分。

在此，引用文献[29]的分叉引理，C 首先与 A 进行一组攻击游戏。A 将输出一个有效的签名 $\sigma=(\kappa,\sigma_1,\sigma_2,\sigma_3,\sigma_4)$。C 以重放的方式和 A 再进行一组攻击游戏，即这组游戏的所有运行条件与之前那组完全相同，唯一不同的是 C 重排了 Hash 函数的请求答复，同时由于 C 模拟生成的 Hash 函数值是随意分布的，以此重排后的值仍然有效且不可区分。经过这一组游戏，A 同样输出一个有效签名 $\sigma=(\kappa,\sigma_1,\sigma_2,\sigma_3,\sigma_4)$。

C 通过 $\sigma=(\kappa,\sigma_1,\sigma_2,\sigma_3,\sigma_4)$ 和 $\sigma=(\kappa,\sigma_1,\sigma_2,\ \sigma_3,\sigma_4)$ 构造一个新的 SDH 对。并进行如下计算：$\Delta\kappa=\kappa\text{-}\kappa'$，$\Delta s_1=s_1-s_1'$，$\Delta s_2=s_2-s_2'$，$\Delta s_3=s_3-s_3'$。

依次计算 R_1、R_2、R_3、R_4、R_5

$$R_1=R_1'(R_1=u^{s_1}C_1^{-\kappa})\Rightarrow u^{s_1}C_1^{-\kappa}=u^{s_1'}C_1^{-\kappa'}\Rightarrow$$

$$u^{s_1-s_1'}=C_1^{\kappa-\kappa'}\Rightarrow u^{\frac{\Delta s_1}{\Delta\kappa}}=C_1$$

$$R_2=R_2'(R_2=u^{s_2}C_2^{-\kappa})\Rightarrow u^{s_2}C_2^{-\kappa}=u^{s_2'}C_2^{-\kappa'}\Rightarrow$$

$$u^{s_2-s_2'}\Rightarrow C_2^{\kappa-\kappa'}\Rightarrow u^{\frac{\Delta s_2}{\Delta\kappa}}=C_2$$

$$R_3=R_3'\left(R_3=\hat{e}(C_2,g_2)^{s_4}\hat{e}(h,W)^{-s_1-s_2}\hat{e}(h,g_2)^{-s_{\delta_1}-s_{\delta_2}}\left(\frac{\hat{e}(C_3,W)}{\hat{e}(g_1,g_2)}\right)^{\kappa}\right)\Rightarrow$$

$$\hat{e}(C_2,g_2)^{s_4}\hat{e}(h,W)^{-s_1-s_2}\hat{e}(h,g_2)^{-s_{\delta_1}-s_{\delta_2}}\left(\frac{\hat{e}(C_3,W)}{\hat{e}(g_1,g_2)}\right)^{\kappa}=$$

$$\hat{e}(C_2,g_2)^{s_4'}\hat{e}(h,W)^{-s_1'-s_2'}\hat{e}(h,g_2)^{-s_{\delta_1}'-s_{\delta_2}'}\left(\frac{\hat{e}(C_3,W)}{\hat{e}(g_1,g_2)}\right)^{\kappa'}\Rightarrow$$

$$\hat{e}(C_2,g_2)^{\Delta s_4'}\hat{e}(h,W)^{-\Delta s_1'-\Delta s_2'}\hat{e}(h,g_2)^{-\Delta s_{\delta_1}'-\Delta s_{\delta_2}'}=\left(\frac{\hat{e}(C_3,W)}{\hat{e}(g_1,g_2)}\right)^{\Delta\kappa'}\Rightarrow$$

$$\hat{e}(g_1,g_2)=$$

$$\hat{e}(C_2,g_2)^{\frac{\Delta s_4'}{\Delta\kappa'}}\hat{e}(h,W)^{-\Delta s_1'-\frac{\Delta s'}{\Delta\kappa'2}}\hat{e}(h,g_2)^{-\Delta s_{\delta_1}'-\frac{\Delta s_{\delta_2}'}{\Delta\kappa'}}$$

$$R_4{=}R_4'(R_4{=}u^{-s_3}C_1^{s_2})\Rightarrow u^{-s_3}C_1^{s_2}=u^{-s_3'}C_1^{s_2'}\Rightarrow$$

$$C_1^{s_2-s_2'}=u^{s_3-s_3'}\Rightarrow C_1^{\Delta s_2}=u^{\Delta s_3}\Rightarrow$$

$$u^{\frac{\Delta s_1}{\Delta\kappa^{\Delta s_2}}}=u^{\Delta_{s3}}\Rightarrow\Delta s_1\Delta\kappa^{-1}\Delta s_2=\Delta s_3$$

$$R_5{=}R_5'(R_5{=}u^{-s_4}C_2^{s_2})\Rightarrow u^{-s_4}C_2^{s_2}=u^{-s_4'}C_2^{s_2'}\Rightarrow$$

$$C_2^{s_2-s_2'}=u^{s_4-s_4'}\Rightarrow C_1^{\Delta s_2}=u^{\Delta s_4}\Rightarrow$$

$$u^{\frac{\Delta s_2}{\Delta\kappa^{\Delta s_2}}}=u^{\Delta s_4}\Rightarrow\Delta s_2\Delta\kappa^{-1}\Delta s_2=\Delta s_4$$

最后得到结果为 $\hat{e}(g_1,g_2)=\hat{e}\left(C_2h^{\frac{\Delta s_2}{\Delta\kappa}},g_2^{\frac{\Delta s_2'}{\Delta\kappa'}}\right)$ 。

定义 $A'{=}C_2h^{\frac{\Delta s_2}{\Delta\kappa}}$ ， $x'=g_2^{\frac{\Delta s_2'}{\Delta\kappa'}}$ ，有 $\hat{e}(g_1,g_2)=\hat{e}(A',x')$ ，这样得到一个新的 SDH 对，也就解决了 SDH 难题，而 SDH 难题为一个困难计算问题，产生矛盾，所以该方案具有不可伪造性。

证毕。

参考文献：

[1] MCKEOWN N. Software-defined networking[C]//IEEE International Conference on Computer Communications. 2009: 30-32.

[2] 王蒙蒙, 刘建伟, 陈杰, 等. 软件定义网络:安全模型、机制及研究进展[J].软件学报,2016,27(4): 969-992.
WANG M M, LIU J W, CHEN J, et al. Software defined networking: security model, threats and mechanism[J]. Journal of Software, 2016, 27(4): 969-992.

[3] AFOLABI I, TALEB T, SAMDANIS K, et al. Network slicing and softwarization: a survey on principles, enabling technologies, and solutions[J]. IEEE Communications Surveys & Tutorials, 2018, 20(3):1.

[4] PORRAS P, SHIN S, YEGNESWARAN V, et al. A security enforcement kernel for OpenFlow networks[C]//The First Workshop on Hot Topics in Software Defined Networks. ACM, 2012: 121-126.

[5] 冯登国, 陈成. 属性密码学研究[J]. 密码学报, 2014, 1(1):1-12.
FENG D G, CHEN C. Research on attribute-based cryptography[J]. Journal of Cryptologic Research,2014,1(1):1-12.

[6] TAKAHASHI N, KODAIRA S, TSURU T, et al. Seismic structure and seismogenesis off Sanriku region, northeastern Japan[J]. Geophysical Journal of the Royal Astronomical Society, 2018, 159(1):129-145.

[7] PORRAS P, CHEUNG S, FONG M, et al. Securing the software-defined network control layer[C]//Annual Network and Distributed System Security Symposium. 2015.

[8] SHIN S, SONG Y, LEE T, et al. Rosemary: a robust, secure, and high-performance network operating system[C]//The 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 78-89.

[9] SHIN S, PORRAS P, YEGNESWARAN V, et al. FRESCO: modular composable security services for software-defined networks[J]. Proceedings of Network & Distributed Security Symposium, 2013.

[10] WEN X, CHEN Y, HU C, et al. Towards a secure controller platform for OpenFlow applications[C]//The Second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking. ACM, 2016: 171-172.

[11] CASADO M, FREEDMAN M J, PETTIT J, et al. Ethane:taking control of the enterprise[C]// ACM Sigcomm Conference on Applications. ACM, 2007:1-12.

[12] 郑鹏, 胡成臣, 李昊. 基于流量特征的 OpenFlow 南向接口开销优化技术[J]. 计算机研究与发展, 2018, 55(s2):346-357.
ZHEN P, HU C C, LI H. Reducing the southbound interface overhead for OpenFlow based on the flow volume characteristics[J]// Journal of Computer Research and Development,2018,55(s2):346-357.

[13] BALLARD J R, RAE I, AKELLA A. Extensible and scalable network monitoring using OpenSAFE[C]// Internet Network Management Conference on Research on Enterprise Networking. USENIX Association, 2010:8.

[14] WUNDSAM A, LEVIN D, SEETHARAMAN S, et al. OFRewind: enabling record and replay troubleshooting for networks[C]// Usenix Conference on Usenix Technical Conference. USENIX Association, 2011:29.

[15] HALPERN E J, PIGNATARO E C. Service function chaining (SFC) architecture[C]// Internet Engineering Task Force. 2015.

[16] 赵志远, 孟相如, 苏玉泽, 等. 多控制器条件下区分 QoS 的虚拟 SDN 映射方法[J]. 通信学报, 2017,38(8):101-110.
ZHAO Z Y, MENG X R, SU Y Z, et al. Virtual SDN embedding with differentiated QoS under multiple controller[J]. Journal on Communication, 2017,38(8):101-110.

[17] 毕军. SDN 体系结构与未来网络体系结构创新环境[J]. 电信科学, 2013, 29(8):6-15.
BI J. SDN architecture and future network innovation environment[J]. Telecommunications Science, 2013, 29(8):6-15.

[18] DARGAHI T, CAPONI A, AMBROSIN M, et al. A survey on the security of stateful SDN data planes[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3):1701-1725.

[19] LU G, SHI Y, GUO C, et al. CAFE: a configurable packet forwarding engine for data center networks[C]// ACM SIGCOMM 2009 Workshop on Programmable Routers for Extensible Services of Tomorrow. DBLP, 2009:25-30.

[20] ATTIG M, BREBNER G. 400 GB/s programmable packet parsing on a single FPGA[C]//IEEE, 2011:12-23.

[21] 金子晋, 兰巨龙, 江逸茗, 等. SDN环境下基于QLearning算法的业务划分路由选路机制[J]. 网络与信息安全学报, 2018, 4(9): 17-22.
JIN Z J, LAN J L, JIANG Y M, et al. QLearning based business differentiating routing mechanism in SDN architecture[J]. Chinese Journal of Network and Information Security, 2018, 4(9):17-22.

[22] PORRAS P, SHIN S, YEGNESWARAN V, et al. A security enforcement kernel for OpenFlow networks[C]//The First Workshop on Hot Topics in Software Defined Networks. ACM, 2012: 121-126.

[23] SHIN S W, PORRAS P, YEGNESWARA V, et al. Fresco: modular composable security services for software-defined networks[C]//20th Annual Network & Distributed System Security Symposium. NDSS, 2013.

[24] 周启钊, 于俊清, 李冬. SDN环境下SAVI动态配置技术研究[J]. 通信学报, 2018, 39(S1):241-249.
ZHOU Q C, YU G Q, LI D. Dynamic source address validation in software defined network[J]. Journal on Communications, 2018, 39(S1):241-249.

[25] KHADER D. Attribute based group signatures[J]. IACR Cryptology ePrint Archive, 2007, 2007: 159.

[26] GOYAL V, PANDEY O, SAHAI A, et al. Attribute-based encryption for fine-grained access control of encrypted data[C]//The 13th ACM Conference on Computer and Communications Security. ACM, 2006: 89-98.

[27] CASADO M, FREEDMAN M J, PETTIT J, et al. Ethane: taking control of the enterprise[C]//ACM SIGCOMM Computer Communication Review. ACM, 2007, 37(4): 1-12.

[28] CASADO M, GARFINKEL T, AKELLA A, et al. SANE: a protection architecture for enterprise networks[J]. USENIX Security Symposium, 2006, 49: 137-151.

[29] PANG R, ALLMAN M, BENNETT M, et al. A first look at modern enterprise traffic[C]//The 5th ACM SIGCOMM Conference on Internet Measurement. USENIX Association, 2005: 2.

[30] BONEH D, BOYEN X, SHACHAM H. Short group signatures[C]// Annual International Cryptology Conference. Springer, 2004: 41-55.

[31] POINTCHEVAL D, STERN J. Security arguments for digital signatures and blind signatures[J]. Journal of Cryptology, 2000, 13(3): 361-396.

[32] REN Y, DING N, WANG T, et al. New algorithms for verifiable out sourcing of bilinear pairings[J]. Science China Information Sciences, 2017, 59(9): 99-103.

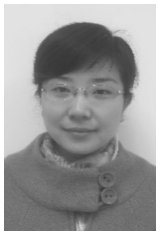
[33] WANG M, LIU J, CHEN J, et al. PERM-GUARD: authenticating the validity of flow rules in software defined networking[C]//International Conference on Cyber Security and Cloud Computing. IEEE, 2017:1-17.

[作者简介]

祝现威（1991− ），男，河南虞城人，信息工程大学博士生，主要研究方向为 SDN 安全、网络安全、云计算安全。

常朝稳（1966− ），男，河南滑县人，博士，信息工程大学教授、博士生导师，主要研究方向为移动信息安全、物联网安全。

朱智强（1961− ），男，河南信阳人，博士，信息工程大学教授、硕士生导师，主要研究方向为云计算安全、信息安全战略、可信计算。



秦晰（1978− ），女，河南焦作人，信息工程大学副教授、硕士生导师，主要研究方向为 SDN 安全、可信计算。