doi:10.11959/j.issn.1000-436x.2016214

基于匿名化流表的网络数据分组实时匿名方法

韩春静[1,2,3]，葛敬国[3]，谢高岗[1]，李亮雄[3]，李佟[3]，刘韵洁[1]
（1. 中国科学院计算技术研究所，北京 100190；2. 中国科学院大学，北京 100049；
3. 中国科学院信息工程研究所，北京 100093）

摘 要：提出了基于匿名化流表的网络数据分组实时匿名方法（Fad-Pan，online trace anonymization based on the anonymous flow table），主要研究 Fad-Pan 算法以及研发基于 DPDK 的 Fad-Pan 原型系统。实验结果表明，Fad-Pan 算法比已有的方法在匿名化速度上提高了 20 倍以上，单个普通服务器可以实时处理万兆链路的 IPv4 和 IPv6 流量数据。

关键词：网络流量匿名化；Fad-Pan；AFT；DPDK
中图分类号：TP391 **文献标识码**：A

Online trace anonymization based on anonymous flow table

HAN Chun-jing[1,2,3], GE Jing-guo[3], XIE Gao-gang[1], LI Liang-xiong[3], LI Tong[3], LIU Yun-jie[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;
2.University of Chinese Academy of Sciences, Beijing 100049, China;
3. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China)

Abstract: A real-time network packet anonymous method named Fad-Pan (online trace anonymization based on the anonymous flow table) was proposed. The Fad-Pan algorithm was studied and an online trace anonymization prototype system based on DPDK library was developed. The experimental results prove that the Fad-Pan algorithm is faster more than 20 times than the existing method, and a single server can handle the real-time IPv4 and IPv6 traffic of the 10 Gbit/s link used by the Fad-Pan.

Key words: network trace anonymization, Fad-Pan, AFT, DPDK

1 引言

真实的互联网流量数据对网络研究及其相关系统研发具有非常重要的价值，网络中任何一种新的算法、协议或体系结构（如 SDN[1]、Openflow[2]、NDN[3]、XIA[4]）在应用或部署前，都必须经过严谨的测试和验证。然而，在模拟仿真环境中，由于节点规模、拓扑、流量种类等原因，基于模拟环境的测试结果和现网的实际应用相比，往往呈现出很大的差异，而且现网中也无法进行大规模测试，所以用大规模的在线/离线网络流量对研究成果进行验证是非常必要的。此外，网络安全研究人员需要最新的网络流量数据，分析流量趋势和用户行为，以便修正已有的安全监测模型；网络安全开发人员也需要 10 Gbit/s 或 100 Gbit/s 等真实链路流量去验证系统的稳定性、准确性等。总之，真实的网络流量数据对网络相关领域的研究等具有很高的价值，尤其是未抽样的原始数据分组。然而，由于真实

收稿日期：2016-05-30；**修回日期**：2016-10-26
基金项目：中国科学院先导专项基金资助项目（No.XDA06010306）；国家自然科学青年基金资助项目（No.F020802）；国家重点基础研究发展计划（“973”计划）基金资助项目（No.2012CB315803）；国家高技术研究发展计划（“863”计划）基金资助项目（No.2013AA013501）
Foundation Items: The Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA06010306), The National Natural Science Youth Foundation of China (No.F020802), The National Basic Research Program of China (973 Program) (No.2012CB315803), The National High Technology Research and Development Program of China (863 Program) (No.2013AA013501)

的网络流量数据涉及用户 IP 地址以及内容等隐私，因此，很少有 ISP 公布这类数据以供研究之用，即使现有已公布的数据，也大多是比较陈旧的流量数据和抽样的网络流，这些数据还存在流量持续时间短、所在链路带宽小等不足，无法满足上述需求。因此，本文的研究对象是高速链路的网络流匿名化技术。

网络流匿名化的实质是对网络数据分组的 IP 地址、载荷等字段采用加密方式进行处理，使处理前后的信息之间的相关性尽可能小，从而达到保护用户 IP 地址和内容等隐私数据不被泄露的目的，此外，加密算法要求尽可能地保持数据分组之间的流量特征，以便相关领域的研究使用。由于运营商链路和数据中心网络的不断升级，10 Gbit/s 或 100 Gbit/s 链路的数据分组采集和实时匿名化难度不断增大，2015 年 3 月，美国 CAIDA 组织在芝加哥的数据中心网络升级到 100 Gbit/s 后，这个采集点的硬件处理能力无法满足分光采集和匿名化而被迫停止[注1]。因此，如何在高速增长的网络带宽下，提高在线网络流量的匿名化速度和扩展性成为当前学术界面临的一大挑战。另外，已有匿名化性能的改进方法都是基于 IPv4 地址的，对于 IPv6 地址其效果不好，甚至不可行。

综上所述，针对已有的网络数据分组匿名化方法处理速度性能不高和扩展性差的问题，本文提出了一种基于匿名化流表的网络数据分组实时匿名（Fad-Pan, online trace anonymization based on the anonymous flow table）方法。本文主要内容包括研究 Fad-Pan 算法和开发基于 DPDK（Intel data plane development kit）的 Fad-Pan 原型系统，具备高性能和良好的扩展性，实现对 IPv4/IPv6 链路网络流量数据的实时匿名化处理。

2 网络流量匿名化相关工作

网络数据分组不同字段的隐私敏感度不同，如 IP 地址和数据分组内容属于高度敏感信息，协议与端口号则属于第二敏感信息，而其他字段属于低敏感信息。因此，在匿名化数据分组时，高敏感字段必须要求匿名化，但是数据分组内容通常不进行匿名化，而是采取将其截取掉的方式，原因如下。首先，虽然数据分组内容对网络协议测试工作非常有用，端到端的长连接需要将用户通信期间的所有数据分组连接起来才能形成网络流，但是，匿名数据分组会导致 IP 信息失真，把大量不保证顺序的 IP 数据分组重组几乎是不可能的，特别是在 10 Gbit/s 或 100 Gbit/s 高速骨干网络上，所以现有的匿名方案基本不会对 IP 数据分组的净载荷进行处理；另外，数据分组内容匿名化后的数据对于研究人员意义不大，而且需要占据大量的存储空间，如视频应用。所以，目前主要研究网络数据分组的 IP 地址匿名化。

目前，主要的匿名化工具有如下几种。TCPdpriv[5]使用 Tcpdump[6]采集的 pcap 格式的流量数据，设计了不同的选项和 0~99 的安全等级，安全等级（level）中 0 最安全，$level_0$ 将不同的地址映射到整数，$level_{99}$ 则对 IP 地址不加任何处理，TCPdpriv 匿名化 IP 地址时，每次都需要查询加密表，因此，处理性能比较低。CANNIE[7]是一个图形化的网络流匿名化工具，支持 NetFlowV_5、V_7 等格式的流量数据，可以对流量数据 8 个字段的信息进行匿名化。Tcpmkpub[8]采用 Crypto-Pan 算法，处理过程中重置时间戳为 0，并重新计算校验和。FLAIM[9]方法与 CANINE 实现基本一致，Anonym[10]集成了各种 IP 地址匿名化方法和可视化工具。PCAPLib[11]工具提供各种数据分组匿名化算法的集成，并提出了 PCAPAnon 方法，PCAPAnon 方法主要是解决不同应用协议数据分组载荷（payload）的匿名化，使用正则表达式的匹配方式对载荷部分信息进行自动匹配和匿名化替换，保持匿名化前后应用类型和信息长度一致。Netshuffle[12]是基于 K 匿名化算法实时的匿名化工具。文献[13]提出了一种匿名化数据的认证工具。

IP 地址匿名化算法主要有切断算法、随机置换算法和前缀保留算法。切断算法将 IP 地址固定的比特位置为 0，只保留剩余的位（典型比特位有 8、16、24），因此，匿名化的结果不可逆转，并且会丢失原有 IP 地址间的关系及路由特性。如果知道这种映射关系是可逆的，则随机置换算法通过随机函数对需要匿名化的 IP 地址产生一对一的映射；否则，TCPurify[5,14]就是这种算法的典型应用。由于映射关系式是完全随机的，因而，匿名化后的 IP 地址丢失了原有 IP 地址间的关系及路由特性。最后，前缀保留算法是最广泛使用的 IP 地址匿名化算法，如果 2 个 IP 地址最长的共同前缀有 K bit，则匿名化后的 IP 地址的最长共同前缀也有 K bit，这种匿名

注1：http://www.caida.org/data/monitors/passive-equinix-chicago.xml。

化能保持地址间的层次关系和路由特性。主要的前缀保留匿名化算法有 TCPdpriv 的 A50 算法[15]和 Crypto-Pan 算法[16]。由于 A50 算法每次 IP 地址匿名化结果不唯一，不能在分布式的环境下并行处理。Crypto-Pan 算法利用 Rijndael[17]加密算法构造的匿名化函数，Crypto-Pan 算法的映射关系与原 IP 地址分配文件出现的先后次序无关，只要密钥 K 相同，不同 IP 地址分配文件中的 IP 地址不会映射到相同的地址。IP 地址前缀保留算法在其他网络中进行了扩展应用，如 EncrIP[18]的 IP 地址匿名化算法，对进入云计算内部的流量，使用概率加密的方法对 IP 地址进行加密和解密，并保持 IP 地址的前缀长度。文献[19]提出 CDN 网络的 IP 地址匿名化方法，由于需要选择最近的 CDN，前缀部分不能参与匿名化，所以加入了部分前缀的游标地址段对主机号进行匿名化。

上述算法对每个 IP 地址都是从其第一个比特开始逐比特计算，实际网络中，运营商骨干链路的数据分组到达速率非常快，需要处理的数据分组非常大，经验证明，上述前缀保留算法无法满足逐个数据分组在线匿名化的需求。在线匿名化优化方法中，文献[20]使用了预存匿名化结果、子树替换和 Top 散列的方法，提供千兆的匿名化性能。文献[21]中提出了 PSC-PA 算法，该算法也采用预存匿名结果的方式，同时对 IP 地址分段进行匿名化加密，提高处理性能。

对于 IP 地址预存的方法，IPv4 的地址空间是 2^{32}，如果将所有的 IP 地址和匿名化后的结果预存到内存中，那么所需的内存空间是 $2\times4\times2^{32}$=32 GB，对普通服务器设备的压力较大。由于匿名化结果的第 i 位只与 IP 地址的前 i−1 位有关系，PSC-PA[21]将 IPv4 地址的 16 位前缀部分和 16 位后缀部分分开进行匿名化，对于 16 位前缀部分，采用 Crypto-Pan 预先计算出匿名化结果，对于 16 位后缀部分，首先采用随机算法 ISAAC[22]生成一个长位串，然后通过位置对应关系将 16 位后缀部分映射到位串，由于 PSC-Pan 算法采用预存的方式，比 Cryto-Pan 算法快了一个数量级，每秒处理分组 25 万个。近些年，匿名化速度改进研究方面集中在预存效率的优化，如文献[23]提出一种快速预存匿名化结果的方法 FRAA，文献[24]提出动态子树调度数据分组匿名化算法 DS-Pan，为更少的内存预存 IP 地址匿名化。然而，一条万兆链路单个方向的数据分组吞吐率一般在每秒百万数据分组左右，因此，上述的这些方法依然不能满足实时处理的需要。

由于隐私性保护和特征保留，上述匿名化算法本质上都是对每个数据分组的源地址以及目的地址的全部或部分进行匿名化处理，会产生大量加密计算的开销，即使采用预存结果的方式，对于 IPv4 地址，其地址空间比较小，具有可行性，但对于 IPv6 地址不具有可行性。由于本研究进行匿名化的对象是 IPv4/IPv6 双栈万兆链路，即使仅预存 IPv4 地址的前 16 位和 IPv6 地址的前 64 位，也需要 $2\times(8\times2^{128}+2\times2^{32})$ byte 存储空间，普通服务器的内存空间无法满足，同时，IPv4 地址的后 16 位和 IPv6 地址的后 64 位依然需要在线匿名化，随着移动网络应用的丰富和 4G 网络的推广，骨干带宽越来越大，现有的这些方法不能进行大带宽流量的实时匿名化处理，仅仅应用于离线匿名化处理。

3　Fad-Pan 算法的数据结构和算法流程

3.1　Fad-Pan 算法基本思想

Fad-Pan 算法全称为基于匿名化流表的网络数据分组实时匿名方法，此方法基于网络流[25]的观点，即同一条网络流（源地址+目的地址）的数据分组会频繁且集中地出现在一条链路上[26,27]，通过在内存中构建一个基于散列结构的匿名化流表，Fad-Pan 算法实时地处理万兆链路的数据分组。相比预存式匿名化方法，Fad-Pan 算法不需要将所有匿名化后 IP 地址提前存储在流表中，而是结合当前处理机的内存资源设置匿名化流表的大小，为了防止流表的冲突增多导致的性能下降，设置了散列表使用率阈值，当散列表的使用率高于阈值，则直接删除当前散列表，重新建立新的散列表，避免维护过期流表项以及流表项替换所带来的开销。

Fad-Pan 算法的主要过程与 Openflow 流表[2]的处理方法相似，Fad-Pan 对每个需要处理的数据分组，首先查找其源地址（目的地址）是否在匿名化流表中，如果不在流表中，调用匿名化函数对源地址（目的地址）匿名化，将匿名化结果插入流表，并用匿名化地址替换原始地址；如果在流表中，则用流表中匿名化地址替换原始地址，最后输出匿名化后的数据分组。

3.2　AFT 流表数据结构

匿名化流表（AFT, anonymization flow table）用于存储 IP 地址匿名化的信息，AFT 是基于已有

且成熟的 Khash[28]结构，以 Hash-Map 的方式实现，主要由流表关键字数组 *AFT_keys*、流表值数组 *AFT_values*、流表大小 *AFT_size*、流表已使用单元数 *AFT_used* 以及流表最大使用率阈值 *AFT_bound* 等构成。*AFT_keys* 的值由网络数据分组的源 IP 地址(目的 IP 地址)通过给定算法生成；*AFT_values* 的值为对应的匿名化后的源 IP 地址（目的 IP 地址）；*AFT_size* 表示 AFT 流表总共的存储单元数量；*AFT_used* 表示当前 AFT 流表中被使用的存储单元的数量，用于计算 AFT 流表的使用率 U_{AFT}，U_{AFT} 为流表当前已使用的单元和流表总容量的比值，如式(1)所示。*AFT_bound* 表示 AFT 流表存储单元的最大使用率，当 AFT 流表的使用率 U_{AFT} 高于或等于 *AFT_bound* 时，需要扩展或者删除当前的流表，其原因在于 AFT 流表基于散列表，因此 AFT 流表的查找和插入等操作所产生的散列冲突会随着 AFT 流表使用率的增大而增大。

$$U_{\mathrm{AFT}} = \frac{AFT\_used}{AFT\_size} \tag{1}$$

AFT_keys 和 *AFT_values* 值的设置有 2 种方式，第一种为地址对（IP pair），*AFT_keys* 的值为源地址+目的地址，*AFT_values* 的值为匿名化后的源地址和目的地址，这种方式的流表可以在输出原始数据分组时保存一些会话 ID；第二种 *AFT_keys* 的值是源地址或目的地址，*AFT_values* 的值为匿名化后源地址或目的地址。经过实验测试发现使用第二种方式的流表时，匿名化的速度更快，而且所需的存储空间仅仅是第一种方式的 $\frac{1}{6}$。2 种流表结构本质上相同，如果单纯考虑匿名化需求，第二种方式的流表在匿名化速率和存储空间上都更优，所以本文主要讨论基于第二种流表的相关算法。图 1 为本文中基于 IPv4 地址的 AFT 流表数据结构。

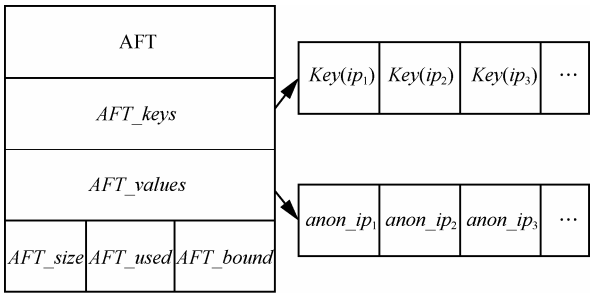


图 1 AFT 流表结构

3.3 AFT 流表扩展

本文研究的实际应用环境是 IPv4/IPv6 双栈万兆链路，同时，需要匿名化处理 IPv4 和 IPv6 这 2 种地址。IPv4 地址和 IPv6 地址的差异比较大（IPv6 为 128 位，IPv4 为 32 位），在 AFT 流表中关键字值的计算等方面，IPv6 地址的耗时也会更多，目前的网络中主要还是以 IPv4 流量为主，AFT 流表中 IPv4 地址匿名化和流表插入等操作频率远远高于 IPv6 地址，如果将 IPv4 地址与 IPv6 地址存储在同一个 AFT 流表中，会影响到整体的性能。因此，针对图 1 中的 AFT 流表，本文研究进行了扩展，扩展后的 AFT 流表由 AFT 流表头节点以及 2 个子流表构成，AFT 流表头节点包含指针 *AFT_P_4* 和 *AFT_P_6*，*AFT_P_4* 指向 IPv4 子流表 AFT4，*AFT_P_6* 指向 IPv6 子流表 AFT6。图 2 所示为扩展后的 AFT 流表结构，其中，AFT4 存储 IPv4 地址匿名化信息，AFT6 存储 IPv6 地址匿名化信息。

3.4 Fad-Pan 算法流程

Fad-Pan 算法流程如下。

先初始化 AFT 流表，然后获取网络数据分组并进行相关处理，解析获取到的网络数据分组，得到数据分组的源/目的地址并查找流表 AFT，如果 AFT 中存在源/目的地址的记录，则取对应记录的匿名化地址，替换数据分组对应的源/目的地址并重新计算校检和；如果 AFT 中不存在源/目的地址的记录，则对源/目的地址进行匿名化，用匿名化后的地址替换数据分组对应的源/目的地址并重新计算校检和，最后将匿名化后的地址作为流表项插入流表中。当 AFT 流表使用率到达或超过阈值上限后，删除当前的 AFT 流表，重新建立新的 AFT 流表，继续处理网络流量数据分组。

其伪代码描述如下。

输入 网络流无限序列 $S = s_0, s_1, \cdots, s_n, n>>\infty$

输出 匿名化后的数据分组

```
begin
AFT_init(AFT); //初始 AFT 化流表
for each s_i∈S do
  Src_IP = Pkt_parse(s_i);  //解析出数据分组的源地址
  Dst_IP = Pkt_parse(s_i);//解析出数据分组的目的地址
  if AFT.use ≥ Max_use  //判断 AFT 流表利用率时候超过给定的阈值
```

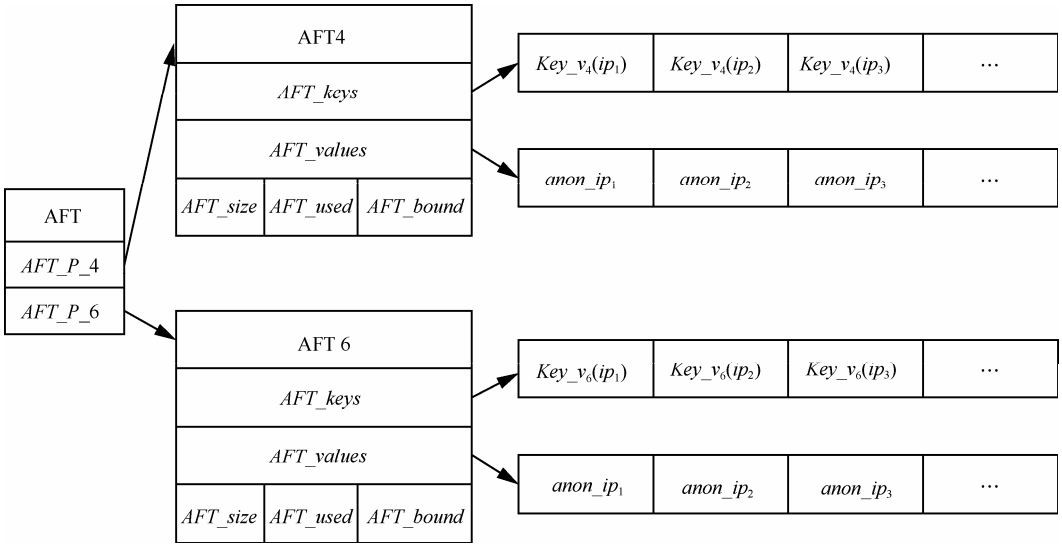


图 2　扩展后的 AFT 流表结构

then //删除当前 AFT 流表，并建立新的流表

AFT_destroy(AFT);

AFT_init(AFT);

else

index = AFT_find(AFT, *Src_IP*);

if index != null

then //流表中存在记录

s_i.*src_ip* = AFT [index].*A_Src_IP*;　//用流表中匿名化后的地址替换数据分组 s_i 源地址

else //流表中不存在记录

A_Src_IP = IP_anon(*Src_IP*); //调用匿名化算法对 *Src_ip* 地址匿名化处理

AFT_insert (AFT, *A_Src_IP*); //将匿名化后的地址插入到 AFT 流表

s_i.*src_ip*= *A_Src_IP*;　//匿名化结果的地址替换源地址

index = AFT_find (AFT, *Dst_IP*);

if index != null

then //流表中存在记录

s_i.*dst_ip* = AFT [index].*A_Dst_IP*;　//用流表中匿名化后的地址替换数据分组 s_i 目的地址

else //流表中不存在记录

A_ Src_IP = IP_anon(*Dst_IP*);//调用匿名化算法对 *Dst_IP* 地址匿名化处理

AFT_insert(AFT,*A_Dst_IP*); //将匿名化后的地址插入到 AFT 流表

s_i.*src_ip*= *A_Dst_IP*;　//匿名化结果的地址替换目的地址

Sendout(s_i);

end

Fad-Pan 算法的主要操作有：AFT 流表初始化（AFT_init），负责流表的创建以及初始化；AFT 流表删除（AFT_destroy），负责删除当前的流表，回收流表占用的内存空间；AFT 流表查找（AFT_find），在 AFT 流表中查找当前所处理的 IP 地址是否已经存在；AFT 流表插入（AFT_insert），将匿名化后的 IP 地址插入到 AFT 流表中；IP 地址匿名化(IP_anon)，调用 IP 地址的匿名化算法处理。

AFT_find 负责查找需要匿名化的 IP 地址是否在 AFT 流表中，若该 IP 地址是 IPv4 地址，则查找 AFT IPV4 子流表；若是 IPv6 地址，则查找 AFT IPv6 子流表，由于 AFT 流表属于散列结构，因此，其查找的时间复杂度是 $O(1)$，如果不计算冲突因素的时间开销，其查找的时间开销可以忽略。对于一个 IP 地址，若 AFT 流表的 *AFT_keys* 中存在此地址，则 AFT_find 查找成功并返回该 IP 地址在 AFT 表中的索引值；否则查找失败，返回空值。

AFT_insert 负责将 AFT 流表中不存在的 IP 地址及其匿名化后的地址插入到 AFT 流表中。若该 IP 地址是 IPv4 地址，则插入到 AFT IPv4 子流表，若是 IPv6 地址，则插入到 AFT IPv6 子流表，其插入的时间复杂度也是 $O(1)$，在不计算冲突的情况下，其时间开销也是可以忽略的。

IP_anon 实现 IP 地址的匿名化处理，输入为 IP

地址，输出其匿名化后的地址。IP_anon 算法基于开源的 Crypto-Pan 算法，改进了 Crypto-Pan 的性能并增加了对 IPv6 地址的匿名化处理的接口，使 Crypto-Pan 支持对 IPv6 地址的匿名化，匿名化之前需要判断 IP 地址的类型，调用不同的匿名化接口。由于 AFT 表中主要存储匿名化之前的 IP 地址和匿名化之后的 IP 地址，这个表与具体的匿名化加密算法无关，所以，AFT 流表结构可以扩展到不同粒度的网络，同时，也支持目前最新的一些算法，如本文第 2 节提到的 EncrIP 算法和 CDN 的匿名化算法。

4 系统实现

4.1 系统架构概述

本文研究基于 Fad-Pan 算法的在线网络流量数据分组 IP 地址匿名化系统，简称 Fad-Pan 系统。Fad-Pan 系统由 DPDK 数据分组采集平台和数据分组匿名化模块组成，如图 3 所示。Fad-Pan 系统通过分光设备，将需要进行匿名化处理网络链路的上、下行 2 条物理链路分别接入到服务器的光网卡。图 3 中，$port_0$ 和 $port_1$ 对应光网卡；DPDK 数据分组采集平台采集数据分组，并将采集到的数据分组通过 RSS[29]插入到网卡的相应队列中；数据分组匿名化模块在 DPDK 平台上设计了多线程模型，处理网卡收分组队列中的数据分组，并将匿名化后的数据分组导出到匿名化文件中。DPDK 采集平台通过 $port_0$ 和 $port_1$ 采集链路的上行和下行流量，$port_0$/$port_1$ 每捕获一个数据分组，通过 RSS 计算出该数据分组二元组（*src_ip*，*dst_ip*）的散列值，然后根据散列值将该数据分组分发到 $port_0$/$port_1$ 中对应的缓存队列，同一会话的数据分组或分发到 $port_0$ 第 *i* 个缓存队列中，或分发到 $port_1$ 第 *i* 个缓存队列中，*i* 为其散列值。

由于不同缓存队列的数据分组由不同的线程处理，同一个会话的数据分组匿名化结果会被输出到不同结果文件中，这样，使用者就无法研究网络会话的整个生命周期[30]。为了保证同一个应用会话的双向数据分组输出到同一个匿名化结果文件中，Fad-Pan 系统修改了 DPDK 数据分组采集平台，将 $port_1$ 的第 *k* 个缓存队列绑定到 Fad-Pan 的 $port_0$ 第 *k* 个处理线程，Fad-Pan 线程 *k* 每次获取 $port_0$ 的第 *k* 个队列的数据分组的同时也获取 $port_1$ 的第 *k* 个队列的数据分组，这样就解决了上述问题。

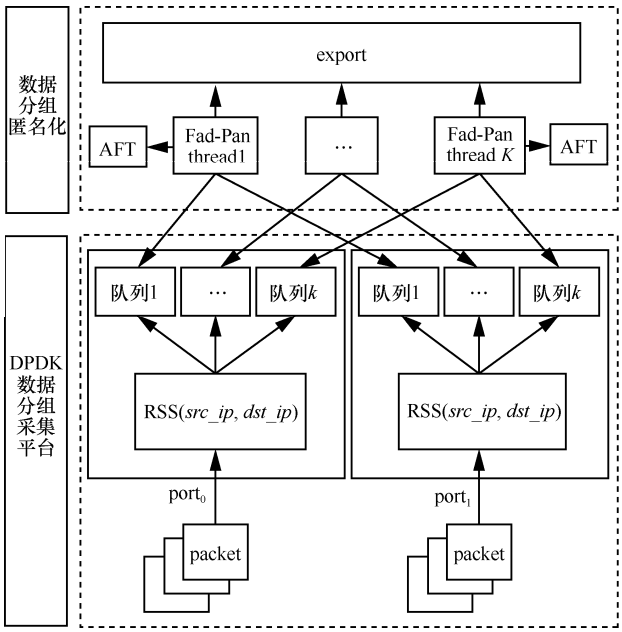


图 3 数据分组在线匿名化系统（Fad-Pan）架构

匿名化模块中 AFT 流表可以设置成全局模式，也可以设置成局部模式。如果 AFT 流表采用全局模式，则所有匿名化线程共享一张全局 AFT 流表，任何 IP 地址在全局 AFT 流表中不会产生重复的记录，但存在流表插入操作需要多线程同步的缺点，实际测试的时候出现了 70%的分组丢失率。如果 AFT 流表采用局部模式，每个匿名化处理线程拥有独立的局部 AFT 流表，则不会产生由于流表插入更新导致的线程同步问题，从而提高匿名化并发处理的速率，但是流表之间会有重复记录，实际性能测试时不影响整体的实时匿名化速度，故采用后一种方法来实现系统。

4.2 系统实现及优化

本文原型系统使用 C 语言进行开发，数据分组采集平台部分重写了 DPDK 中 rte_eth_rx_burst 等相关代码，使其满足本系统的需要，本文原型系统中通过 DPDK 设置处理服务器单个网卡的接收队列数量为 *queue*=8，同时启用 8 个线程处理网卡中对应的 8 个队列中的数据分组，可以扩展设置队列数量；AFT 流表采用局部模式，每个线程独立拥有一张独立的局部 AFT 流表。匿名化模块部分的代码开发基于已成熟的 Crypto-Pan、Khash、OpenSSL 等，其中，IP_anon 匿名化代码基于 Crypto-Pan 以及 OpenSSL[31]中的 aes-ni[32]，AFT 流表以及流表操作功能基于 Khash。

实现系统的过程中，为了提高处理速度，优化

了 Fad-Pan 算法以及整个系统，主要包括以下几点。

1) 由于 Khash 结构的散列表大小默认是动态扩展的，当其利用率超过给定的阈值时需要扩展，需要事先指定每次扩展后的大小。每次扩展后，由于散列表大小发生变化，因此，所有表项的索引也发生变化，需要重新计算所有表项的索引值，从而带来很大的时间开销。对于骨干网，数据分组到达速度非常快，流表会迅速地扩张，如果采用 AFT 流表动态扩展的策略，会产生大量的时间成本，为了避免这个问题，本文采用了固定 AFT 流表，针对 IPv4 和 IPv6，分别设置其 *AFT_size* 为 S_4 和 S_6。

为了实现快速查找，AFT 流表需要常驻内存，因此，S_4 和 S_6 的大小必须考虑到所使用的处理服务器的内存大小情况，本文中设置了 AFT 内存最大使用量为 M，M 的取值可以根据经验进行分配，假如给定了 M 的取值，则有

$$12S_4 + S_6 = M \tag{2}$$

IPv4 子流表大小与 IPv6 子流表大小之间需满足

$$S_4 = CS_6 \tag{3}$$

其中，C 表示所处理的链路上一定时间内的不重复 IPv4 与不重复 IPv6 地址数量的比值，上述关系保证了 2 个子流表之间能够平等地利用内存资源。本文 C 值是根据离线网络流量数据计算出的，具体计算方式是以小时为统计单位，统计了同一条链路上大约一周的流量数据中 IPv4 和 IPv6 不相同地址的数量比例，并计算出一系列统计结果的平均值，得到最终的 C 值。由于当前网络中 IPv6 流量所占比例比较小，因此，C 的值很大，S_6 的值很小，以至于 IPv6 子流表频繁地删除与新建，为了解决这一问题，本文采用调节因子 α 对式(2)进行修正，修正后结果见式(4)，α 默认为 1，当 α 值小于 1 时，可以调节增大 S_6 的值，当 α 值大于 1 时，可以调节减小 S_6 的值。

$$S_4 = \alpha CS_6 \tag{4}$$

2) 传统的流表管理方法根据流的生存和活跃时间来删除过期的流表项，需要在流表中增加时钟计时，并且需要定期监控，称为流表项维护策略，这种策略增加了流表处理的开销，因此，Fad-Pan 算法中没有采用这种流表管理方法，在 Fad-Pan 流表管理方法中，当 AFT 流表的使用率高于最大使用率时，便删除当前 AFT 流表，新建 AFT 流表，这称为流表一次性删除策略，虽然 AFT 初始化后必然造成流表项命中率降低，但是根据实验测试，采用流表一次性删除策略时系统的整体性能好于采用流表项逐项维护策略时系统的整体性能。为了分析不同的 AFT 流表管理策略对于匿名化系统整体性能的影响，本文使用第 5 节中的 isp_1-IPv4 数据集进行了相关实验，如图 4 和图 5 所示。图 4 为匿名化每百万数据分组的处理时间开销，并定义处理时间开销=流表操作时间+数据分组匿名化操作时间。通过图 4 可以看出，刚开始 2 种策略都需要进行流表初始化，因此，处理百万数据分组的时间开销都比较大，随着时间的推移，流表一次性删除策略不用维护 AFT 流表中信息，故采用流表一次性删除策略处理百万数据分组的时间开销明显小于采用流表项维护策略的百万分组处理时间开销，虽然图 4 中流表一次性删除策略在重建流表后有抖动，但是可以看出抖动时间范围很窄，抖动的时间小于 2 次流表重建时间间隔的 5%，对整体性能影响很小；相反，流表项维护策略随着时间推移处理百万数据分组的时间越来越大，通过分析还发现，增加的时间开销主要包括 AFT 流表项维护以及替换老化流表项的操作。

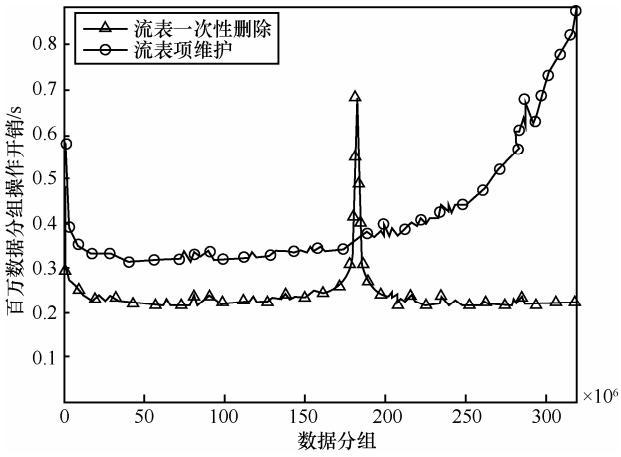


图 4　2 种流表管理策略下操作时间开销对比

图 5 为处理数据分组过程中上述 2 种策略对应的流表使用率，当流表利用率达到阈值时，流表一次性删除策略删除 AFT 并重建，可以看出图 4 中的抖动处与图 5 中的 AFT 重建点重合；在刚开始处理数据分组时，2 种策略的 U_{AFT} 增长率保持一致，随着时间推移，流表项维护策略开始检查并替换掉老化表项，其 U_{AFT} 增长率低于流表一次性删除策略的 U_{AFT} 增长率，而且持续一段时间后其利

用率接近于 100%。虽然流表项维护策略由于 U_{AFT} 流表空间利用率高可以减少非匹配项，但是维护、替换等其他开销导致了其处理百万数据分组的时间开销大于流表一次性删除策略处理百万数据分组的时间开销。

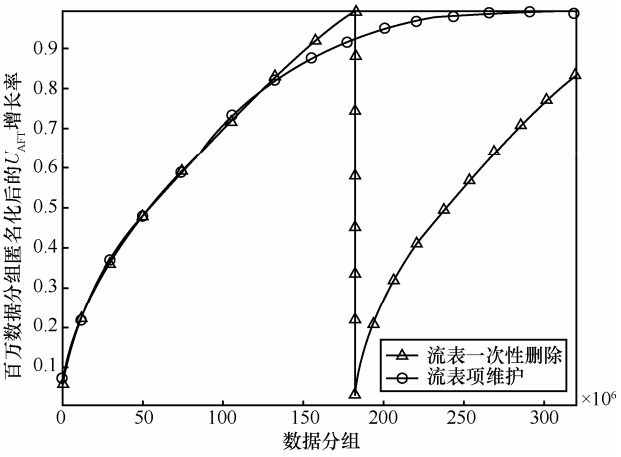


图 5 2 种流表管理策略下流表利用率对比

除此之外，本文还优化调整了 AFT 流表的最大使用率，一般 AFT 流表的使用率越大，表明目前所匿名化的网段中不相同的 IP 地址比较多，或者是当前 AFT 流表大小设置的相对比较小，因此会导致 AFT 流表的冲突概率越大；同时，流表利用率越小，则重建流表的频率越大，AFT 流表最大使用率需要折中处理。起初 *AFT_bound*（流表利用率）设置为流表大小的 $\frac{3}{4}$ 左右，即 *AFT_bound*=77%，本文通过实验发现当 AFT 流表项是单地址结构时，流表命中率非常高。若 *AFT_bound*=77%，则会频繁地删除和重建 AFT 流表，造成的开销远大于流表产生冲突的开销，通过实验还发现，当 *AFT_bound*=100%，即使流表的冲突发生率最高，但整体的开销代价是最小的，因此，本文设置 *AFT_bound*=100%。

3) AFT 流表起初设计的时候，*AFT_keys* 的值采用点分形式的 IP 地址字符串，一条 IPv4 流表项占用 19 byte 内存空间（*keys*+*vals* = 15 + 4 = 19 byte），一条 IPv6 流表项占用 55 byte 内存空间（*keys*+*vals* = 39 + 16 = 55 byte），可以看出 IPv6 流表项占用空间比较大，实验同时发现由于 *AFT_keys* 字符串过长造成流表匹配的速度较慢。优化改进后，*AFT_keys* 的值采用整型数据类型，一条 IPv4 流表项占用 8 byte 内存空间（*keys*+*vals* = 4 + 4 = 8 byte），一条 IPv6 流表项占用 32 byte 内存空间（*keys*+*vals* = 16 + 16 = 32 byte），由于流表项关键字值变短，匹配速度大约提高了 40%，并且节约了大约 50%的存储空间。

针对匿名化系统进行了如下优化。

首先，实验原型中发现，匿名化过程涉及到数据分组的拷贝需要重新开辟内存空间，会产生大量的分组丢失。经过优化后，接收到数据分组后直接用匿名化结果替换原始分组的原有地址，由于数据分组 IP 地址替换开销可以忽略不计，因此，大大提高了处理性能。

其次，优化了匿名化函数 IP_anon，本文调用 OpenSSL 的 aes-n$_i$ 接口实现对 IP 地址的加密。通过测试发现，对初期的匿名化速度提升非常明显，测试结果显示，同样处理三百万数据分组，使用 OpenSSL 中的 Rijndael 算法和使用 Crypto-Pan 原始的算法代码整体分组吞吐量相差 6 倍，提升了匿名化处理中的 IP 地址加密速度。

最后，系统对处理线程数量、AFT 流表结构和内存空间利用率进行优化后，单个流表的内存使用量在 150 MB~0.5 GB，因此，总共流表最大使用内存量 8 GB，普通的万兆双核 Intel 处理器，32 GB 以上的内存可以满足其性能。

5 实验结果与分析

本文实验中数据分组在线匿名化系统 Fad-Pan 的服务器配置为 x86 双核 CPU 和 128 GB 内存，DPDK 设置为 8 个数据分组处理线程，需要 8 个 AFT 流表，AFT 内存最大使用量设置成 128 GB×10%；AFT 流表采用局部模式，*C* 值为 100；α 系数设置为 0.07，S_4 和 S_6 结果是 89 478 485 和 17 895 697。

实验数据来自一条运营商万兆骨干网链路和数据中心万兆出口。运营商实验数据的采集时间段为 2015 年 1 月和 3 月白天和晚上忙时各 2 h，1 月份数据和 3 月份数据的实验结果基本一致，本文仅取 3 月份的结果进行展示。实验数据 1 采集于一条 IPv4 骨干链路，称之为 isp$_1$-IPv4，实验数据 2 采集于一条 IPv4/v6 双栈链路，称之为 isp$_2$-IPv4（白天）、isp$_3$-IPv4（晚上）、isp$_4$-IPv6（白天）、isp$_5$-IPv6（晚上），数据中心实验数据称之为 idc-IPv4。

本文对比了 Fad-Pan 和 Crypto-Pan 算法在对上述数据集进行 IP 地址匿名化的处理效率等，Crypto-Pan 属于基于分组级别的匿名化算法，需要对每个数据分组进行匿名化加密处理，Fad-Pan 属于基于流级别的匿名化算法，每条流匿名化加密处

理第一个分组，后续分组只需要查找流表替换。文献[21]中提到的 IP 地址预取的方法在实验过程中发现对 IPv6 地址不适用，故不在比较之列。为了更精准地对比不同算法的处理效率，匿名化时间只统计了匿名化处理过程中的耗时，数据分组获取、数据分组存储的耗时不做计算。

表 1~表 4 为 4 组 IPv4 数据地址匿名化的时间结果。实验中采用 4 组 IPv4 数据集，isp_1-IPv4、isp_2-IPv4、isp_3-IPv4 和 idc-IPv4，每个数据集处理其前 3 亿个数据分组；实验中采用的 4 种算法分别为 Fad-Pan-128、Fad-Pan-256、Crypto-Pan-128 和 Crypto-Pan-256，其中，Fad-Pan-128、Crypto-Pan-128 和 Fad-Pan-256、Crypto-Pan-256 分别指相应算法中采用的加密密钥长度为 128 位和 256 位。加密密钥越长，则生成的匿名化结果越不容易被还原，但是所带来的耗时代价越大。

表 1　　isp_1-IPv4 匿名化时间比较

加密密钥长度/位	匿名化时间/s		加速比
	分组级别 Crypto-Pan	流级别 Fad-Pan	
128	91.87	5.31	17.30
256	210.94	7.30	28.90

表 2　　isp_2-IPv4 匿名化时间比较

加密密钥长度/位	匿名化时间/s		加速比
	分组级别 Crypto-Pan	流级别 Fad-Pan	
128	135.56	7.22	18.78
256	205.85	10.48	19.65

表 3　　isp_3-IPv4 匿名化时间比较

加密密钥长度/位	匿名化时间/s		加速比
	分组级别 Crypto-Pan	流级别 Fad-Pan	
128	140.01	6.31	22.20
256	205.90	8.77	23.49

表 4　　idc-IPv4 匿名化时间比较

加密密钥长度/位	匿名化时间/s		加速比
	分组级别 Crypto-Pan	流级别 Fad-Pan	
128	142.50	2.53	56.27
256	205.55	3.20	64.23

从表 1~表 4 可以看出，匿名化处理相同的 IPv4 数据集 Fad-Pan 算法所耗费时间明显比 Crypto-Pan 算法少很多；Fad-Pan-128 算法和 Fad-Pan-256 算法所耗费的时间基本上差别不大，相比之下，Fad-Pan-128 算法耗费时间更少，Fad-Pan-256 算法对 IP 地址的保护性更好，因此，实际使用时可以根据处理速率和安全性的综合需求选择加密密钥长度；Crypto-Pan-128 算法与 Crypto-Pan-256 算法所耗费的时间相差很大，这也充分反映出基于流级别的 Fad-Pan 在速率和安全性上的兼顾。另外，Fad-Pan 算法匿名化处理数据集 isp_1-IPv4、isp_2-IPv4、isp_3-IPv4 耗费的时间相差不大，相对 Crypto-Pan 的加速比，大概在 20 倍，处理数据集 idc-IPv4 时耗时最少，相对于 Crypto-Pan 算法其加速比在 56 倍以上，可以看出 idc-IPv4 数据对于 Fad-Pan 算法的敏感度最高，经过对数据分组地址对的分析，发现原因在于 idc-IPv4 数据集中的地址相对集中，反映在真实网络中，则表明访问 idc 的用户更加集中。对于 IPv4 的前 3 个数据集，使用了文献[18]的 EncrIP 匿名化方法，使用 Fad-Pan 和普通的逐分组匿名化，加速比也在 20 倍左右，进一步说明基于流级别的 Fad-Pan 带来的性能提升与网络流密切相关，比如网络流的命中率等。

表 5 和表 6 为 2 组 IPv6 数据地址匿名化的时间结果。实验中采用的 2 组 IPv6 数据集分别为 isp4-IPv6 和 isp5-IPv6，每个数据集处理前 3 亿个数据分组；实验中采用的 4 种算法分别为 Fad-Pan-128、Fad-Pan-256、Crypto-Pan-128 和 Crypto-Pan-256。

表 5　　isp_4-IPv6 匿名化时间比较

加密密钥长度/位	匿名化时间/s		加速比
	分组级别 Crypto-Pan	流级别 Fad-Pan	
128	401.02	7.10	56.50
256	575.97	7.26	79.29

表 6　　isp_5-IPv6 匿名化时间比较

加密密钥长度/位	匿名化时间/s		加速比
	分组级别 Crypto-Pan	流级别 Fad-Pan	
128	471.13	10.28	45.84
256	640.59	10.76	59.56

从表 5 和表 6 可以看出，与 IPv4 数据的处理结果相似，匿名化处理相同的 IPv6 数据集 Fad-Pan 算法所耗费时间明显比 Crypto-Pan 算法少很多；Fad-Pan-128 算法和 Fad-Pan-256 算法所

耗费的时间基本上差别不大，Crypto-Pan-128 算法与 Crypto-Pan-256 算法所耗费的时间相差却很大。

另外值得关注的是，通过表 1~表 6 中 Fad-Pan 算法处理 6 个数据集各自前 3 亿个数据分组的结果可以看出，在处理 IPv6 数据集时所耗的时间跟处理 IPv4 数据集时所耗的时间相差不大，且 IPv6 相对加速比更大，甚至处理 isp_4-IPv6 所用时间比处理 IPv4 的某些数据集所用时间更少，为此本文更进一步统计了 IPv6 和 IPv4 地址离散度的情况，发现 IPv6 的地址范围集中度更高，原因在于目前 IPv6 用户和应用都比较集中，因此，其 AFT 命中率更高，如图 6 所示。这也解释了为何 Fad-Pan 算法处理 isp4-IPv6 数据时效果更好。

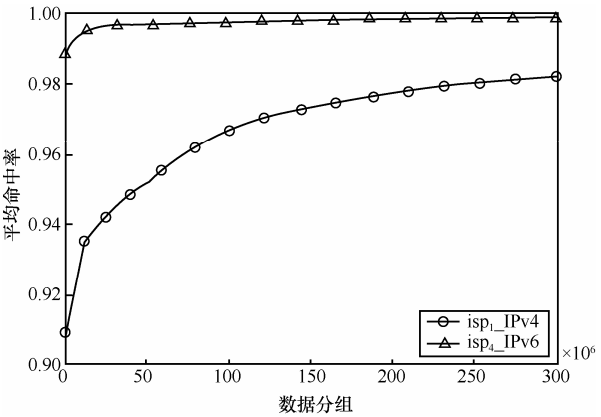


图 6 地址命中率统计

图 6 为 Fad-Pan 算法处理上述 6 个数据集时，每一百万数据分组统计一次结果，得到的 IPv4 地址和 IPv6 地址在 AFT 中的命中率，可以看出 IPv6 地址的命中率始终高于 IPv4 地址的命中率。

通过上述实验比较，本文基于 AFT 流表的 Fad-Pan 数据分组匿名化算法明显优于其他分组匿名化的算法，而且不管是处理 IPv4 地址还是 IPv6 地址，长度为 128 位的密钥和 256 位的密钥对 Fad-Pan 算法性能影响很小，因此，本文建议尽量使用长度为 256 位的密钥，因为兼具安全性和匿名化速度。

本文第 3 节详细介绍了 AFT 流表，影响 AFT 流表处理效率的因素主要有 AFT 流表自身的冲突产生的查找代价以及删除 AFT 流表后新建流表的代价。流表利用率的设置需要平衡这 2 种代价，使对匿名化整体效率的提升最大化。本文将通过实验分析流表利用率对匿名化处理速率的影响。

本文对 isp_2-IPv4 数据集进行了实验，实验中分别设置 AFT 流表的最大利用率为 77%和 100%，并进行相应的统计，为了方便统计，实验时每一百万数据分组统计一次处理时间，图 7 为单线程下匿名化处理时间的统计结果，可以看出 2 种 AFT 利用率下，第一个百万数据分组的处理时间都比较耗时，由于刚开始需要建立 AFT 流表，会产生大量的流表查找，插入操作，之后每百万个数据分组的处理时间都大幅减少且比较均衡，网络流局部性的特征[33]在广域网也有所体现，图 7 右边部分曲线的值突然波动增大，其原因在于删除原 AFT 后，新建 AFT 流表比较耗时；通过图 7 还可以看出，AFT 流表利用率设置为 100%时没有引起 Fad-Pan 算法处理性能的下降，和 AFT 流表利用率设置为 77%时 Fad-Pan 算法处理性能几乎一致；相反，AFT 流表利用率设置为 77%时，AFT 流表由于重建导致性能下降。因此说明流表最大利用率设置为 77%时，可以减少流表的冲突率，但是流表重建频率会增大；流表最大利用率设置为 100%时，虽然流表冲突率会增加，但是流表重建的频率会越小，从实验中可以发现流表重建的代价更大。

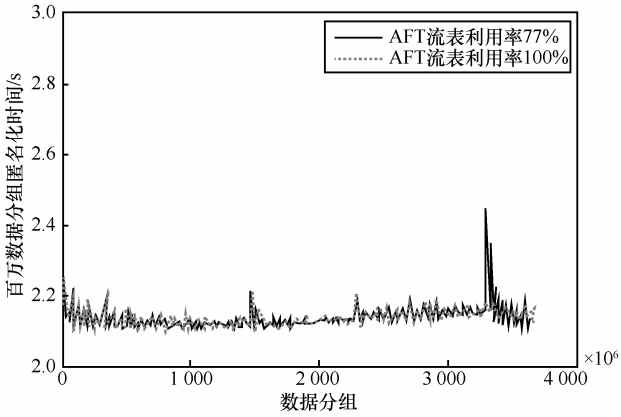


图 7 IPv4 2 种流表利用率下的匿名化时间比较

6 结束语

最新的网络原始流量数据对研究和研发人员都非常重要，由于涉及用户隐私信息，因此，匿名化原始网络数据必不可少，结合目前需求，已有的匿名化方法和系统无法满足实时在线匿名化数据分组 IP 地址的需求，本文提出了基于网络匿名化流表的数据分组实时匿名化方法 Fad-Pan，研究了其相关的算法以及研发了 Fad-Pan 系统原型，对 IPv4

和 IPv6 的最新运营商流量都进行了详细的测试和原型性能验证，证明此方法的可行性；原型系统基于 x86 普通服务器架构，具有很好的扩展性。本系统目前在中科院先导项目中试用，为中科院海云创新实验平台[34]（scie.ac.cn）提供实验数据源。未来的工作主要包括流表设计的改进，提高流量处理的性能；改进流表的设计，能够支持更多的网络地址，如 MPLS 等，待完善后将公开源代码，做好维护工作。

参考文献：

[1] JAIN S, KUMAR A, MANDAL S, et al. B4: experience with a globally-deployed software defined WAN[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 3-14.

[2] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. Openflow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[3] ZHANG L, ESTRIN D, BURKE J, et al. Named data networking (NDN) project[J]. Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC, 2010.

[4] HAN D, ANAND A, DOGAR F R, et al. XIA: efficient support for evolvable internetworking[C]//Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12). 2012: 309-322.

[5] PANG R, ALLMAN M, PAXSON V, et al. The devil and packet trace anonymization[J]. ACM SIGCOMM Computer Communication Review, 2006, 36(1): 29-38.

[6] FUENTES F, KAR D C. Ethereal vs tcpdump: a comparative study on packet sniffing tools for educational purpose[J]. Journal of Computing Sciences in Colleges, 2005, 20(4): 169-176.

[7] LI Y, SLAGELL A, LUO K, et al. Canine: a combined conversion and anonymization tool for processing netflows for security[C] //International Conference on Telecommunication Systems Modeling and Analysis. 2005: 21.

[8] PANG R, ALLMAN M, PAXSON V, et al. The devil and packet trace anonymization[J]. ACM SIGCOMM Computer Communication Review, 2006, 36(1): 29-38.

[9] SLAGELL A J, LAKKARAJU K, LUO K. FLAIM: a multi-level anonymization framework for computer and network logs[C] //LISA. 2006, 6: 3-8.

[10] FARAH T, TRAJKOVIC L. Anonym: a tool for anonymization of the Internet traffic[C] //IEEE International Conference on Cybernetics (CYBCONF). IEEE, 2013: 261-266.

[11] LIN Y D, LIN P C, WANG S H, et al. Pcaplib: a system of extracting, classifying, and anonymizing real packet traces[J]. IEEE Systems Journal, 2016, 10 (2): 520-531.

[12] PAUL R R, VALGENTI V C, KIM M S. Real-time net shuffle: graph distortion for on-line anonymization[C]//The 2011 19th IEEE International Conference on Network Protocols. IEEE Computer Society, 2011: 133-134.

[13] DINKAR K P, JAIN S A. Security and privacy preserving policy in mobile crowd sensing[J]. International Journal of Engineering Science, 2016, 5206.

[14] BLANTON E. Tcpurify: TCP packet sniffer[EB/OL]. (2002-9). http://irg.cs. ohiou. edu/eblanton/tcpurify.

[15] BELLARE M. Practice-oriented provable-security[J]. Lecture Notes in Computer Science, 1998, 156(11): 221-231.

[16] FAN J, XU J, AMMAR M H, et al. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme[J]. Computer Networks, 2004, 46(2): 253-72.

[17] DAEMEN J, RIJMEN V. AES proposal: Rijndael[S]. US: NIST, 2003.

[18] NATARAJAN S, WOLF T. Network-level privacy for hosted cloud services[C]//2014 International Conference and Workshop on the Network of the Future (NOF), IEEE, 2014: 1-8.

[19] LI S, DOH I, CHAE K. An anonymous IP-based privacy protection routing mechanism for CDN[C]//2016 International Conference on Information Networking (ICOIN). 2016: 75-80.

[20] RAMASWAMY R, WOLF T. High-speed prefix-preserving IP address anonymization for passive measurement systems[J]. IEEE/ACM Transactions on Networking, 2007, 15(1): 26-39.

[21] 史冰, 吴连国, 丁伟. IP 地址前缀保留匿名化算法的改进[J]. 微电子学与计算机, 2007, 24(10):167-170.
SHI B, WU L G, DING W. Improvement of prefix-preserving IP address anonymization algorithm[J]. Microelectronics & Computer, 2007, 24(10): 167-170.

[22] JENKINS J R J. Isaac[C] //Fast Software Encryption. Springer Berlin Heidelberg, 1996: 41-49.

[23] ZHANG P, HUANG X, LUO M, et al. Fast restorable prefix- preserving IP address anonymization for IPv4/IPv6[J]. The Journal of China Universities of Posts and Telecommunications, 2010, 17: 93-98.

[24] CAO K, LI Y, YANG H, et al. Poster: an online prefix-preserving IP address anonymization algorithm for passive measurement systems[C]//International Conference on Security and Privacy in Communication Systems. Springer International Publishing, 2015: 581-584

[25] AHUJA R K, MAGNANTI T L, ORLIN J B. Network flows[R]. School of Management Cambridge, MA, 1988.

[26] THOMPSON K, MILLER G J, WILDER R. Wide-area Internet traffic patterns and characteristics[J]. Network, 1997, 11(6): 10-23.

[27] 张广兴, 张大方, 谢高岗, 等. Internet 城域出口链路流量测量与特征分析[J]. 电子学报, 2007, 35(11): 2092-2097.
ZHANG G X, ZHANG D F, XIE G G, et al. Internet traffic measurement and characteristic analysis on output link of metro area network[J]. Acta Electronica Sinica, 2007, 35(11): 2092-2097.

[28] BAKSHI R, CUDRE M P, WYLOT M. A comparison of different data structures to store RDF data[R]. 2013.

[29] GOGLIN S D, CORNETT L. Flexible and extensible receive side scaling: U.S. Patent 7,584,286[P]. 2009-9-1.

[30] WOO S, PARK K. Scalable TCP session monitoring with symmetric receive-side scaling[R]. Korea: KAIST, 2012.

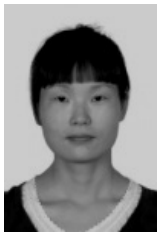
[31] VIEGA J, MESSIER M, CHANDRA P. Network security with openssl: cryptography for secure communications[M]. O'Reilly Media, Inc, 2002.

[32] KROYETZ T, ROGAWAY P. The software performance of authenticated-encryption modes[C]//Fast Software Encryption. Springer Berlin Heidelberg, 2011. 306-327.

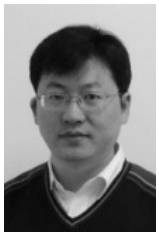
[33] ALMEIDA V, BESTTAYROS A, CROYELLA M, et al. Characterizing reference locality in the WWW[C]// Fourth International Conference on. Parallel and Distributed Information Systems. IEEE, 1996: 92-103.

[34] 葛敬国, 唐海娜, 鄂跃鹏, 等. 海云创新试验环境管控与服务系统总体设计[J]. 网络新媒体技术, 2012, 1(6): 45-51.
GE J G, TANG H N, E Y P, et al. The overall design of resource control and service system in the sea-cloud innovative and experimental environment[J]. Journal of Network New Media, 2012, 1(6): 45-51.

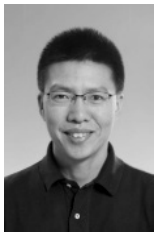
作者简介：



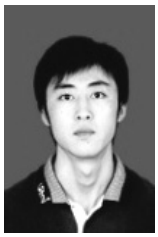
韩春静（1978-），女，河南郑州人，中国科学院博士生，中国科学院高级工程师、硕士生导师，主要研究方向为网络测量与行为分析、网络信息流识别与处理等。



葛敬国（1973-），男，安徽肥东人，博士，中国科学院研究员、博士生导师，主要研究方向为网络体系结构与安全防护、网络测量与行为分析。



谢高岗（1974-），男，浙江东阳人，博士，中国科学院研究员、博士生导师，主要研究方向为软件定义网络、网络功能虚拟化、互联网体系结构、网络测量分析。



李亮雄（1986-），男，甘肃会宁人，中国科学院工程师，主要研究方向为网络测量分析。

李佟（1978-），男，辽宁盘锦人，中国科学院高级工程师、硕士生导师，主要研究方向为网络和计算机体系结构、网络测量与行为分析。

刘韵洁（1943-），男，山东烟台人，中国科学院院士，中国科学院教授、博士生导师，主要研究方向为未来网络架构及关键技术、网络融合与演进。