支持数据隐私保护的恶意加密流量检测确认方法

何高峰[1]，魏千峰[1]，肖咸财[1]，朱海婷[1]，徐丙凤[2]
（1. 南京邮电大学物联网学院，江苏 南京 210003；2. 南京林业大学信息科学技术学院，江苏 南京 210042）

摘 要：为解决基于机器学习的恶意加密流量检测易产生大量误报的问题，利用安全两方计算，在不泄露具体数据内容的前提下实现网络流量内容和入侵检测特征间的字符段比对。基于字符段比对结果，设计入侵检测特征匹配方法，完成关键词的精准匹配。为保证所提方法的有效执行，提出用户终端输入随机验证策略，使恶意用户终端难以使用任意数据参与安全两方计算进而躲避检测确认。对所提方法的安全性和性能进行了理论分析，并采用真实部署和仿真实验相结合的方式进行验证。实验结果表明，所提方法能显著提升检测效果，且资源消耗低。

关键词：恶意加密流量；机器学习；安全两方计算；自动确认
中图分类号：TP39
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2022034

Confirmation method for the detection of malicious encrypted traffic with data privacy protection

HE Gaofeng[1], WEI Qianfeng[1], XIAO Xiancai[1], ZHU Haiting[1], XU Bingfeng[2]
1. School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
2. College of Information Science and Technology, Nanjing Forestry University, Nanjing 210042, China

Abstract: In order to solve the problem that excessive false positives in the detection of encrypted malicious traffic based on machine learning, secure two-party computation was used to compare character segments between network traffic and intrusion detection rulers without revealing the data content. Based on the comparison results, an intrusion detection feature matching algorithm was designed to accurately match keywords. A random verification strategy for users' input was also proposed to facilitate the method. As a result, malicious users couldn't use arbitrary data to participate in secure two-party calculations and avoid confirmation. The security and resource consumption of the method were theoretically analyzed and verified by a combination of real deployment and simulation experiments. The experimental results show that the proposed method can significantly improve the detection performance with low system resources.

Keywords: malicious encrypt traffic, machine learning, secure two-party computation, automatic confirmation

0 引言

在互联网、物联网以及工业互联网的发展和演化中，网络流量加密日益被广泛接受和采用，成为一项重要的网络通信安全保护机制[1-2]。但网络流量加密技术是一把双刃剑，在保护正常用户网络通信

收稿日期：2021−11−16；**修回日期**：2022−01−19
通信作者：徐丙凤，bingfengxu@njfu.edu.cn
基金项目：国家自然科学基金资助项目（No.61802192, No.61702282）；南京邮电大学校级自然科学基金项目（No.NY221096）；南京航空航天大学基本科研业务费科研基地创新基金资助项目（No.NJ2020022）
Foundation Items: The National Natural Science Foundation of China (No.61802192, No.61702282), The Natural Science Foundation of Nanjing University of Posts and Telecommunications (No.NY221096), The Fundamental Research Funds for the Central Universities, NUAA (No.NJ2020022)

安全的同时，也可以被攻击者所利用以隐藏攻击特征，从而躲避检测。为此，现有研究提出多种基于机器学习的恶意加密流量检测方法。基于机器学习的检测方法可以仅使用报文方向、报文长度和报文时间间隔等基本元素特征，不需要查看流量内容，因而被视为一种自然的恶意加密流量检测手段[3]。Anderson 等[4]分析了 18 个恶意软件家族产生的上万条恶意传输层安全协议（TLS, transport layer security）加密流量，并利用逻辑回归模型进行分类。实验结果表明，分类的准确率超过 98%。除了使用逻辑回归等基本机器学习算法外，研究人员还结合最新的深度学习算法进行恶意加密流量检测。Wang 等[5]首先将网络流量字节转变为图像，然后使用卷积神经网络进行建模分类，平均准确率超过 99%。

在实际应用中，基于机器学习的恶意加密流量检测易产生大量误报[6]。以科罗拉多大学博尔德分校校园网络为例[7]，该网络中共包含 33 000 余名学生和 7 000 余名教职工，每小时产生大约 1 000 万条 TLS 流量。若检测系统的误报率为 0.01%（文献[4]中的典型结果），则每小时产生 1 000 条误报，一天则多达 24 000 条误报。这些误报若均由人工分析排除，将是非常艰难的工作。同时，主流的深度学习方法大多为黑箱运作模式，难以解释说明一条加密网络流量由于存在何种特征而被判断为恶意流量，从而无法为网络管理人员提供可靠的分析依据。这进一步增大了以人工方式排除误报的困难性。

为解决上述难题，一种可能的思路是将特征匹配和机器学习相结合，从而实现误报的自动排除。如文献[8]中所述，在长达 16 个月的实际测试中，基于特征匹配的入侵检测系统共产生 1 800 余条误报。其中，85%的误报与网络管理策略相关，如对等网络（P2P, peer to peer）流量识别等，仅有 270 余条误报与实际网络攻击相关。因此，可通过设置合适的入侵检测特征对机器学习算法的判别结果进行确认，从而减少误报数量，且匹配的特征能够作为网络管理人员的分析依据。为此，一种直观的实现方法可描述为针对加密网络流量，检测节点（如网络中部署的入侵检测系统）首先利用机器学习算法进行恶意性判别，若判断为恶意流量，通知用户终端如电脑、智能手机等将本地保存的会话密钥发送至检测节点；然后进一步利用该密钥对流量进行解密，获得明文内容；最后在流量明文内容上开展入侵检测特征匹配，若匹配成功，则确认当前流量为恶意流量，否则为误报。

上述特征匹配和机器学习的简单结合又将引发新的网络安全问题。例如，检测节点可宣称任意的加密网络流量为恶意流量，要求用户终端提供对应会话密钥，进而获得明文通信内容，威胁用户的网络通信隐私安全。或者由用户终端解密网络流量，检测节点将入侵检测特征发送至用户终端处进行匹配。但同样地，该方法无法保护检测节点的数据隐私（商用入侵检测特征一般需要付费购买）。同时，恶意用户终端可以任意宣称匹配结果，进而躲避检测。

本文提出一种安全的恶意加密流量检测确认方法，旨在保护数据隐私的前提下，通过对机器学习方法产生的检测结果进行自动确认，减少误报并产生具体的检测判断依据。为保护数据隐私，由用户终端解密网络流量内容，用户终端和检测节点通过运行安全两方计算协议[9]完成网络流量内容和入侵检测特征间的精准匹配。在此过程中，用户终端和检测节点仅交互加密后的数据，且数据不需要解密处理，实现了用户终端和检测节点的双向隐私保护。特别是，本文考虑并解决恶意用户终端任意输入问题，即恶意用户终端以任意数据作为安全两方计算协议的输入，从而躲避特征匹配过程。本文的主要贡献如下。

1) 提出恶意加密流量检测确认方法。基于安全两方计算协议，在不泄露数据内容的前提下，实现入侵检测字符串关键词、正则表达式关键词以及关键词位置信息的精准匹配，并解决了安全两方计算协议在实际应用中存在的任意输入问题[10]。

2) 对方法的安全性和运行时所消耗的系统资源进行理论分析。给出恶意用户终端成功躲避检测的概率计算式，并指出影响用户终端计算量、所占内存大小和网络带宽消耗的关键因素。

3) 原型系统实现与验证。通过真实系统部署和仿真实验相结合的方式，验证方法功能和性能。实验结果表明，所提方法能显著减少误报，且对网络和系统性能影响小。

1　相关工作

本节对恶意加密流量的现有检测方法进行总结梳理。整体上，现有检测方法可以归纳为基于密文解密、基于密文匹配以及基于机器学习这三类。

1) 基于密文解密的检测方法。此方法是现阶段一种常用的恶意加密流量检测方法，其基本思路简洁明了，首先对加密流量进行解密得到明文内容，然后利用入侵检测特征匹配明文内容。若匹配成功，则判断当前加密流量为恶意流量。TLS 透明代理[11]是这类方法的典型实现。透明代理通常充当中间人（MITM, man-in-the-middle）代理，将加密的网络连接分为两部分：客户端到代理和代理到应用服务器。代理首先将其根证书导入客户的受信任证书存储区。当客户端应用程序建立 TLS 连接到应用服务器时，代理代替应用服务器完成 TLS 握手过程。同时，代理与应用服务器建立第二个 TLS 连接。此后，代理可以任意解密通信内容，并判断 2 个连接之间的消息内容是否为恶意。透明代理易于实施和部署，但也破坏了端到端的加密防护。研究人员已经发现多起 TLS 透明代理有意窃取用户隐私信息的恶意攻击事件[12]。

为克服透明代理的弊端（安装完成后，用户对代理的运行情况一无所知），研究人员设计出非透明代理，如 mcTLS[13]、PlainBox[14]等。mcTLS 是第一个针对 TLS 的非透明代理设计，能提供基于证书的身份认证机制，使用户和应用服务器能够对中间人代理进行身份确认。为实现此功能，mcTLS 需要设计新的握手协议，并需改写现有用户端和服务器端的代码实现，因而在现阶段难以推广使用。与 mcTLS 不同，PlainBox 使用带外方式进行身份认证和密钥共享，不需要对现有 TLS 的协议设计和代码实现进行改动。但与透明代理类似，PlainBox 等非透明代理依然可以读取用户和应用服务器之间的所有通信内容，给用户的隐私保护带来巨大威胁。

2) 基于密文匹配的检测方法。为在检测恶意加密流量的同时实现用户隐私信息保护，研究人员设计了一种新的检测思路：对入侵检测特征进行加密，再将加密后的检测特征与加密流量内容进行直接匹配。该思路在文献[15]中首次被提出。文献[15]实现了一种 BlindBox 系统，该系统由用户终端、应用服务器以及检测节点组成。首先，用户终端与应用服务器建立正常 TLS 连接。在传输数据时，用户终端需要对数据进行 2 次处理。一是通过正常 TLS 连接，将数据发送至应用服务器端。二是对数据进行分割处理，比如将数据分割为多个长度为 5 B 的片段，称之为 token。接着，用户终端利用对称加密算法，如 AES 和密钥 k 对所有的 token 进行加密，并将加密后的 token 和加密算法对应的混淆电路发送至检测节点。检测节点对入侵检测特征进行同样的处理，如分割为长度为 5 B 的 token，然后利用混淆电路形式的 AES 算法和密钥 k 对规则 token 进行加密。最后，检测节点对加密后的流量内容 token 和检测特征 token 进行比对匹配。若匹配成功，则表明对应的加密流量含有恶意内容，为恶意流量。在此过程中，用户的所有数据都为密文形式，检测节点无法得知其具体内容（同规则 token 匹配的内容除外），因而有效保护了用户隐私。但其弊端是给用户终端性能带来严重影响[15]，如分析处理 20 条网络流量，需要消耗 1 003.38 GB 网络带宽，其中包括加密 token 的发送和携带密钥 k 的 AES 加密算法的不经意传输。

在文献[15]的启发下，研究人员设计出多种改进的密文匹配检测方法，如 PrivDPI[16]、SHVE+[17]等。PrivDPI 对 BlindBox 中的规则加密进行优化，使加密后的规则可以多次重复使用，减少了系统初始化配置所花费的时间。但 PrivDPI 生成密文形式的流量内容 token 时，速度是 BlindBox 的 $\frac{1}{6}$，且同样需要对数据进行 2 次处理，严重降低了终端的运行性能。SHVE+对隐藏向量加密（HVE, hidden vector encryption）方案[18]进行改进，使其支持特征匹配功能。在使用时，用户端不需要对数据进行多次处理，只需要在传输至应用服务器时进行 HVE 即可。但 HVE 方案目前并不被典型的网络安全协议如 TLS 等支持，无法部署于实际的网络应用中。

3) 基于机器学习的检测方法。机器学习被视为恶意加密流量检测的一种自然手段[3,19]。这是由于网络流量加密仅改变了报文内容的形式（由明文变为随机字段），并没有显著改变报文方向、报文长度以及报文时间间隔等特征，而这些特征将能有效区分恶意与正常流量。文献[4]针对 TLS 流量，以报文长度序列、报文长度间隔序列、报文字节分布等为特征值，利用逻辑回归算法分类识别恶意 TLS 流量，识别准确率超过 98%。文献[20]针对加密的安全外壳（SSH, secure shell）协议流量，以网络流中的报文数量、访问记录数为特征，实现 SSH 协议流量的入侵检测。文献[21]针对数据的加密窃取，以网络协议、应用层协议以及时间等为特征值建立

网络用户行为模型，并使用多项式朴素贝叶斯分类算法实现加密数据泄露的检测。文献[22]利用聚类算法，从 TLS 流量中提取报文数量、流总字节数、流时长、报文时间间隔均值与方差等作为特征值，实现恶意 Android 应用的在线检测。

为提高检测的准确率以及克服需要人工选择检测特征的难题，文献[5]将恶意软件产生的网络流量转变为图像，然后使用图像处理中成熟的深度学习卷积神经网络进行建模分类，平均准确率超过 99%。类似地，文献[23]使用树形深度神经网络对恶意流量进行分类检测，实验结果表明，检测的准确率为 99.63%，且能较好检测未知的恶意流量。文献[24]给出了基于深度学习的恶意加密流量检测研究综述。在上述工作中，均需要事先采集大量标注的恶意加密流量样本作为训练数据集，且标注正确与否直接影响最终的检测准确性[25]。为此，文献[26]利用生成对抗网络，从少量标注的恶意流量样本中自动合成新的样本，进而提高机器学习算法的分类准确性。

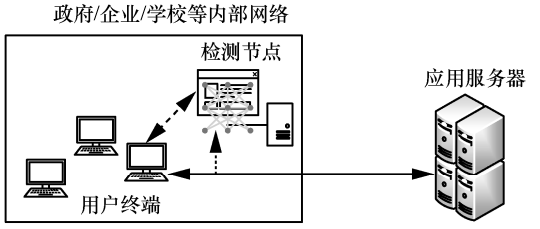
综上所述，基于密文解密的检测方法易于实现和部署，但其本质上破坏了网络应用端到端的加密防护，严重威胁用户数据隐私安全。基于密文匹配的检测方法能较好保护用户的通信隐私，但其性能较低，目前仍处于前期理论研究和探索阶段。基于机器学习的检测方法具有高检测率，且最新的深度学习方法能够从网络流量中自动提取检测特征和自动合成训练数据，无疑增加了此类检测方法的实用性。但基于机器学习的检测方法易产生大量误报，限制了其在实际网络中的部署应用。为此，本文提出一种检测结果再确认方法。在本文提出的方法中，组合使用机器学习和特征匹配功能来实现恶意加密流量检测的可解释和低误报特性，并通过安全两方计算协议确保双方数据的隐私保护。本文工作为恶意加密流量检测提供了一种新的实现思路。

2　方法概述与攻击者模型

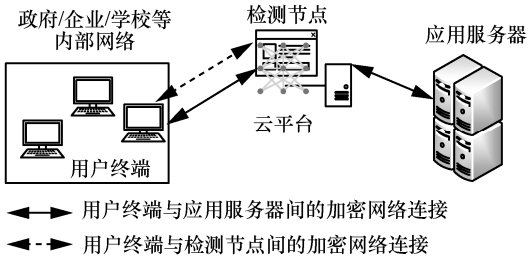
2.1　方法概述

方法的典型部署应用如图 1 所示。用户终端位于网络内侧，通过加密网络连接访问远程应用服务器。对于加密网络连接，用户终端记录网络五元组信息<源 IP 地址、目的 IP 地址、源端口、目的端口、安全协议>以及对应的会话安全参数。其中，安全协议为具体使用的网络通信安全协议，如 TLS、SSH 协议等。会话安全参数为当前网络连接使用的随机数和密钥，如 TLS 中的客户端随机数和预备主密钥。值得注意的是，目前主流的杀毒软件已具备上述功能，因而可以对现有杀毒软件功能进行扩充以支持方法的运行。

为保护网络安全，网络管理员部署检测节点，对网络发出或接收的所有加密流量进行分析，判断是否为恶意流量。检测节点可以为独立硬件设备部署于网络出口处（图 1(a)），或采用外包形式[27]部署于云平台中（图 1(b)）。检测节点所采用的分析方法可以为任意机器学习方法，如逻辑回归[4]、深度学习[5]等，同时配备入侵检测特征库用于检测结果确认。



(a) 检测节点为独立硬件，部署于网络出口处



(b) 检测节点为外包形式，部署于云平台中

图 1　方法的典型部署应用

为对检测结果进行确认，检测节点需与用户终端建立网络连接。在图 1(a)中，用户终端与检测节点位于同一网络中，易于满足该要求。在图 1(b)中，网络管理员可将检测节点所在 IP 地址统一配置于用户终端中，用户终端启动时便与检测节点建立相应网络连接。在后文中，均假定用户终端和检测节点间已存在安全的加密网络连接，所有用户终端构成集合 U。恶意加密流量检测确认流程如图 2 所示。

在图 2 中，检测节点利用机器学习算法对加密网络流量进行恶意性判别。对于识别出的恶意加密流量，

将流量报文发送至对应用户终端（用户终端不需要事先保存流量，从而减轻用户终端处负载），用户终端验证该流量的真实性和隐私性。具体地，用户终端使用<源 IP 地址、目的 IP 地址、源端口、目的端口、安全协议>确定对应的会话安全参数，并解密流量。解密成功，则验证了流量真实性。随后，用户终端验证流量内容中是否含有与自身隐私信息相关内容，如“简历投递”“疾病”等。若无相关信息，则验证通过；否则验证失败。若真实性或隐私性验证失败，用户终端终止确认流程，并记录出错信息便于后续人工分析。

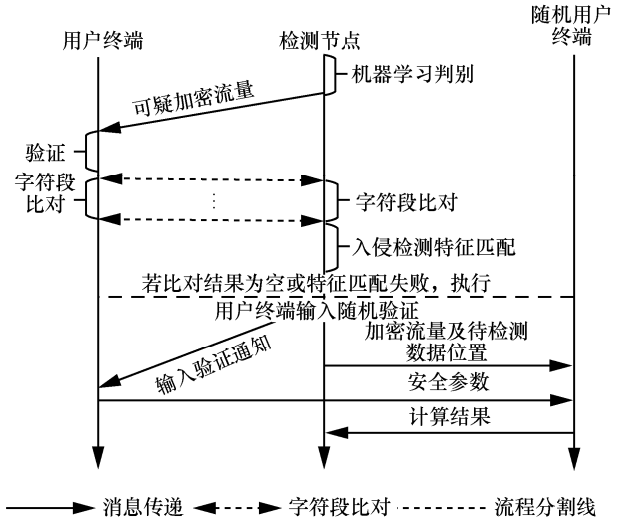


图 2 恶意加密流量检测确认流程

验证通过后，用户终端和检测节点运行安全两方计算协议进行数据安全匹配。具体地，本文采用隐私保护集合求交（PSI, private set intersection）协议[9]，判断网络流量内容和入侵检测特征之间是否存在相同字符片段。检测节点依据求出的交集还原匹配出具体的入侵检测特征。若匹配成功，则确认当前流量为恶意流量；若匹配失败（如交集为空），则启动用户终端输入验证流程，由随机选择的其他用户终端对 PSI 协议数据和流量内容进行随机比对验证。若验证结果正常，表明当前流量为正常加密流量，机器学习检测算法的判别结果为误报；否则，表明用户终端实施了输入替换，存在恶意行为。用户终端输入随机验证使恶意用户终端难以使用其他数据代替真正的流量内容参与 PSI 协议以躲避检测确认。

2.2 攻击者模型

如 2.1 节所述，方法执行主要涉及用户终端和检测节点。考虑实际情形，本文做出如下攻击者模型假定。

检测节点为半诚实模型[15,28]，其严格按照既定流程执行，但可能在运行过程中试图获取用户隐私信息。如图 1 所示，检测节点为网络管理员部署，实现网络安全防护是其核心目标，执行既定流程有利于其实现该功能。但与此同时，检测节点可通过方法的执行窃取用户隐私信息，如推断流量内容中是否含有“工作查询”“简历投递”“疾病”等信息，进而调查、解雇对应员工[29]。因而在检测结果确认过程中，应保护用户的隐私信息不被泄露。

用户终端区分为恶意用户终端和正常用户终端。恶意用户终端为网络入侵者所占据的终端，而正常用户终端为其他合法终端。合理假定网络入侵者已拥有所占据终端的管理员权限，在终端中可执行一切操作，因而此类用户终端为恶意模型，即恶意用户终端能以任意偏离方法的执行流程来躲避检测确认或窃取有用信息。恶意用户终端之间可以相互合谋以完成特定攻击目标。正常用户终端为半诚实模型[15]，在流程执行过程中可以试图获取入侵检测特征的具体内容。用户终端与检测节点间相互独立，检测节点无法控制用户终端以执行恶意行为，如数据窃取等。

3 方法实现

在本文所提方法中，机器学习判别可以由任意机器学习算法完成，并非本文的研究重点。本节将详细阐述字符段比对、入侵检测特征匹配和用户终端输入随机验证的具体实现流程。论文涉及的主要参数符号如表 1 所示。

3.1 字符段比对

本文提出利用入侵检测特征对机器学习算法的判别结果进行进一步确认。入侵检测特征可以利用现有的 ET Pro Ruleset、Snort Talos Rules 等入侵检测规则。这些规则中包括的字符串关键词（如“-J-jar -J”）、正则表达式关键词（如“/(launch\x28.+-J-jar -J|-J-jar -J.+launch\x28)/i”）以及关键词在流量内容中的位置信息即为入侵检测特征。实用的入侵检测系统一般以字符串关键词为先导[30]，即首先匹配字符串关键词，若匹配成功，再匹配对应的正则表达式关键词，从而提高运行效率。本文方法遵循同样思路实现。

表 1　参数符号

参数	含义
U	用户终端集合，其集合元素用 u 表示
C	用户终端处的数据集合，其集合元素用 c 表示
S	检测节点处的数据集合，其集合元素用 s 表示
$\lvert\bullet\rvert$	集合大小
$a \xleftarrow{R} A$	从集合 A 中随机选择元素 a
p, q	p 和 q 为素数，且 $q \mid p-1$
Z_q	小于 q 的正整数集合
$\mathbb{G}, g$	$\mathbb{G}$ 为循环群，g 为 $\mathbb{G}$ 的一个生成元
$H_1()$	随机预言 $H_1()$：$\{0,1\}^* \to \mathbb{G}$
$H_2()$	随机预言 $H_2()$：$\mathbb{G}\times\mathbb{G}\times\{0,1\}^* \to \{0,1\}^k$
ZK{}	离散对数零知识证明
R	随机数，并以上下标区分不同随机数
f	加密网络流量
k_f	流 f 对应的加密密钥
L	数据总长度，并以下标区分不同数据类型
l	数据分割长度
r	攻击者修改的字符数量
e	验证时选择的字符段数量
h	入侵检测特征关键词总数量
λ	资源消耗，并以下标区分不同资源消耗

为实现在特征匹配过程中同时满足用户终端和检测节点的数据隐私保护要求，首先使用 PSI 协议[9]实现数据内容的字符段比对。将网络流量内容和入侵检测关键词分割为多个长度为 l 的字符片段，分别构成集合 C 和 S。针对集合 C 和 S，用户终端和检测节点执行隐私保护集合求交协议得出交集 $C \bigcap S$。在此过程中，除了交集 $C \bigcap S$ 以外，用户终端和检测节点并不能获得对方的其他任何数据内容信息，从而实现了隐私保护。根据交集 $C \bigcap S$，检测节点将从多个字符段中还原出对应的数据内容，从而完成入侵检测特征匹配。本节主要阐述字符段比对的具体技术实现，入侵检测特征匹配将在 3.2 节中描述。

1) 用户终端预处理。用户终端利用保存的会话密钥对可疑的加密网络流量进行解密（会话密钥保存以及加密网络流量获取等内容见 2.1 节），从而获得网络流量明文内容（十六进制形式）。考虑到商用的入侵检测关键词中大多含有十六进制内容，因而在预处理步骤中，将网络流量内容和入侵检测关键词统一转变为十六进制形式，以便后续的集合求交运算。对流量明文内容进行滑动窗口分割，形成集合 C。具体步骤如下。

步骤 1　设定大小为 l ($l \geqslant 1$)的滑动窗口。

步骤 2　从流内容的起始处读取窗口内所有内容，按序加入集合 C。

步骤 3　将窗口后移一位，读取窗口内所有内容，按序加入集合 C。

步骤 4　重复步骤 3，直至窗口内容长度小于 l。

2) 检测节点预处理。检测节点读取入侵检测特征库中的所有字符串形式关键词，将关键词内容转变为十六进制形式并进行滑动窗口分割，形成集合 S。具体步骤如下。

步骤 1　设定大小为 l 的滑动窗口。

步骤 2　读取当前关键词。

步骤 3　从关键词的起始处读取窗口内所有内容，加入集合 S。

步骤 4　将窗口后移一个字符，读取窗口内所有内容，加入集合 S（重复内容删除，不需要保持元素顺序）。

步骤 5　重复步骤 4 直至窗口内容长度小于 l。

步骤 6　跳转至步骤 2，直至所有字符串形式关键词读取完毕。

3) 检测节点加密集合 S。完成预处理后，检测节点加密集合 $S=\{s_1, s_2, \cdots, s_{|S|}\}$ 中每一项元素，并发送至用户终端，即针对 s_j ($1 \leqslant j \leqslant |S|$)，进行式(1)～式(3)的 Hash 和模幂运算。

$$hs_j = H_1(s_j) \tag{1}$$

$$R_s^j \xleftarrow{R} Z_q \tag{2}$$

$$M_j = hs_j(g)^{R_s^j} \bmod p \tag{3}$$

计算完成后，检测节点将集合$\{M_j\}$发送至用户终端。

4) 用户终端加密集合 C。用户终端接收集合$\{M_j\}$后，对 M_j 和集合 $C=\{c_1, c_2, \cdots, c_{|C|}\}$ 进行密码学处理，计算过程如式(4)～式(8)所示。其中，式(4)产生随机数 R_c，式(5)和式(6)利用 R_c 进行对应的模幂运算，式(7)利用离散对数难题和 Hash 计算生成 R_c 的零知识证明，式(8)对集合 C 中的每一项元素进行模幂运算。

$$R_c \xleftarrow{R} Z_q \tag{4}$$

$$Z = (g)^{R_c} \bmod p \tag{5}$$

$$M_j^{'} = (M_j)^{R_c} \bmod p, 1 \leqslant j \leqslant |S| \tag{6}$$

$$\pi = \mathrm{ZK}\{R_c \mid Z = (g)^{R_c} \bmod p, \forall j, M_j^{'} = (M_j)^{R_c} \bmod p\} \tag{7}$$

$$hc_i = H_1(c_i), K_c^i = (hc_i)^{R_c} \bmod p,$$
$$T_c^i = H_2(K_c^i, hc_i, c_i), 1 \leqslant i \leqslant |C| \tag{8}$$

计算完成后，用户终端将 Z、$\{M_j^{'}\}$、$\{T_c^i\}$和 π 发送至检测节点。

5) 检测节点比对数据。检测节点验证零知识证明 π 的值，如果验证失败，流程终止；如果验证通过，进行式(9)计算。式(9)对集合 S 中每一项元素进行模幂和 Hash 处理。

$$K_s^j = (Z)^{-R_s^j} M_j^{'} \bmod p,$$
$$T_s^j = H_2(K_s^j, hs_j, s_j), 1 \leqslant j \leqslant |S| \tag{9}$$

集合 C 和 S 的交集元素计算式为

$$s_j \in C \cap S \text{ if } \exists_{j,i} \text{ s.t. } T_s^j = T_c^i \tag{10}$$

同时，为准确匹配检测特征，记录 T_c^i 所对应的位置信息，即上标 i 的值。由于 C 中元素可重复，存在多个 T_c^i 对应一个 T_s^j 情形，因此需要分别记录。最终的交集集合可表示为

$$\{< s_j \in C \cap S, i >\} \tag{11}$$

式(7)中 π 值的验证参见文献[9, 31]。上述过程为文献[9]中 PSI 协议的优化，主要区别在于检测节点加密集合 S 时不需要提供零知识证明。这是由于本文假定检测节点为半诚实模型（2.2 节），其严格执行协议流程，因此不需要额外证明。

3.2 入侵检测特征匹配

检测节点依据集合 $\{< s_j \in C \cap S, i >\}$ 匹配具体的入侵检测特征。若集合 $\{< s_j \in C \cap S, i >\}$ 不为空，字符串形式关键词匹配算法如算法 1 所示。

算法 1 字符串形式关键词匹配算法

输入 集合 $\{< s_j \in C \cap S, i >\}$ 和参数 para；para 默认为入侵检测特征库

输出 集合{<s_j, i, 匹配结果：成功或失败>}

1) 统计集合 $\{< s_j \in C \cap S, i >\}$ 元素个数，记为 n
2) j := 1
3) while $j \leqslant n$ do
4) if s_j 的后(l−1)个字符同 s_{j+1} 的前(l−1)个字符相同 do
5) 将 s_{j+1} 的最后一个字符内容复制至 s_j 的尾部
6) 从集合中删除 s_{j+1} 对应的元素，并将后续元素均前移一位
7) n := n−1
8) else j := j+1
9) end if
10) end while
11) 从更新后的集合中提取 s_j 的值，并同 para 中所有字符串关键词进行逐一匹配
12) return {<s_j, i, 匹配结果：成功或失败>}

在算法 1 中，若 s_j 匹配成功且对应的入侵检测规则中含有正则表达式关键词，则继续执行算法 2。

算法 2 正则表达式关键词匹配算法

输入 正则表达式关键词，记为 X

输出 匹配结果：成功或失败

1) 检测节点生成 X 的所有实例，实例集合记为 I_X
2) 对 I_X 中每一元素，进行字符分割，即执行检测节点预处理的步骤 1～步骤 6
3) 对应新形成的集合 S，执行式(1)～式(3)，并将集合$\{M_j\}$发送至用户终端
4) 用户终端执行算式(6)和式(7)，并将 Z 和 $\{M_j^{'}\}$发送至检测节点
5) 检测节点执行算式(9)和式(10)，求出交集 $\{< s_j \in C \cap S, i >\}$
6) 以 $\{< s_j \in C \cap S, i >\}$ 和 I_X 为输入，调用算法 1
7) return 匹配结果：成功或失败

入侵检测特征中的关键词位置信息可由更新后的集合 $\{< s_j \in C \cap S, i >\}$ 中 i 和 s_j 的长度直接推算出。如 ET Pro rules 中的 offset 字段，表示关键词在流内容的起始位置，因而 $(i-1)l$ 的值等于 offset 字段的值即匹配成功。其余表示位置信息的字段，如 distance、within 等均可做类似计算和匹配。

在算法 1 中，只需要遍历交集集合 $\{< s_j \in C \cap S, i >\}$ 一次，因而时间复杂度为 $O(n)$。在算法 2 中，步骤 1)～步骤 3)可离线完成（如系统启动时完成），从而加快算法执行速度；步骤 4)～步骤 7)的时间复杂度与正则表达式关键词中适配符数量等因素相关。设共有 x 个正则表达式关键词，每个关键词中含有 a 个适配符，适配符的取值范围为 b，关键词实例长度为 L_I，则集合 S 的大小为 $|S| = (L_I - l + 1)xa^b$。在式(10)中，利用 Hash 表求解集合交集[32]，此时算法遍历 S 集合两次、C 集合一

次以及交集 $C\cap S$ 一次，故时间复杂度为 $O(2(L_I-l+1)xa^b+v+n)$。算法 2 的时间复杂度随适配符数量呈多项式增长。

3.3 用户终端输入随机验证

当式(10)中的交集 $C\cap S$ 为空，或特征匹配失败时，检测节点执行用户终端输入随机验证流程。这是由于在式(8)中，恶意用户终端可以使用任意数据代替流量内容 c_i 的值进行计算，从而规避检测确认。用户终端输入随机验证流程表述如下。

检测节点从 Z_q 中选择随机数 R_d，计算参数 P_d 为

$$P_d=(g)^{R_d}\bmod p \tag{12}$$

检测节点将参数 P_d 和“输入验证通知”发送至目标用户终端。记目标用户终端为 u_t，即 U 集合中第 t 个用户。u_t 接收相关信息后，从 Z_q 中选择随机数 R_t，计算参数 P_t 为

$$P_t=(g)^{R_t}\bmod p \tag{13}$$

用户终端 u_t 将参数 P_t 发送至检测节点。检测节点和用户终端 u_t 分别进行式(14)和式(15)计算，求出共同参数 k，并计算式(16)，求得参数 z（z 应不等于 t）。式(12)～式(15)本质上为 DH 密钥交换[33]，$H_1(k)$为随机数，因而 u_z 为随机选择的用户终端。

$$k=(P_t)^{R_d}\bmod p \tag{14}$$

$$k=(P_d)^{R_t}\bmod p \tag{15}$$

$$z=H_1(k)\bmod |U| \tag{16}$$

$$R_1,\cdots,R_e\overset{R}{\leftarrow}[1,v] \tag{17}$$

$$\widetilde{hc_i}=H_1(C[R_i]),\widetilde{K_c^i}=(hc_i)^{R_c}\bmod p,$$
$$\widetilde{T_c^i}=H_2(K_c^i,hc_i,c_i),1\leqslant i\leqslant e \tag{18}$$

如式(17)所示，检测节点从[1,v]中随机选择 e 个数字 $R_1,\cdots,R_e$，并将 $R_1,\cdots,R_e$ 和加密流量 f 发送至用户终端 u_z。用户终端 u_t 发送会话密钥 k_f 和式(4)中的 R_c 至 u_z。用户终端 u_z 接收 $R_1,\cdots,R_e$、f、k_f 和 R_c 后，利用 k_f 解密流 f。接着，对于解密后的流内容，执行用户终端预处理的步骤 1～步骤 4，得出集合 C。从 C 中选择第 $R_1,\cdots,R_e$ 个元素，利用 R_c 计算式(18)。最后，u_z 将 $\widetilde{T_c^i}$ $(1\leqslant i\leqslant e)$发送至检测节点。

检测节点将 $\widetilde{T_c^i}$ $(1\leqslant i\leqslant e)$同用户终端 u_t 发送的 $\{T_c^i\}$（即式(8)的计算结果）中的对应数值进行比较。若比较结果不一致，则表明 u_t 实施了恶意输入代替，为恶意用户终端。在此过程中，恶意用户终端成功通过验证的概率分析见第 4 节。

4 方法分析

本节对所提方法进行理论分析，包括方法的安全性分析以及性能分析。在安全性分析中，重点分析数据隐私保护特征以及恶意用户终端成功通过验证的概率。性能分析主要针对用户终端进行，分析用户终端需要执行的计算量、使用的内存大小以及消耗的网络通信带宽。类似的性能分析方法同样适用于检测节点，且检测节点一般由专用服务器实现，故本节省略检测节点的资源消耗分析过程。

4.1 安全性分析

1) 数据隐私保护特征分析

本文提出的方法能提供良好的双向数据隐私保护功能：①用户终端无法获得检测节点的入侵检测特征内容；②除交集 $C\cap S$ 的内容外，检测节点无法获知用户终端其他数据内容。方法的隐私保护功能以离散对数问题[34]为理论基础。在式(3)中，由于离散对数的难解性，用户终端无法在多项式时间内从 M_j 反推出 hs_j 的值，进而无法获知式(1)中 s_j 的值，从而实现了检测节点的入侵检测特征隐私保护。类似地，针对式(8)，检测节点无法反推出 c_i 的值，因而除了交集 $C\cap S$ 内容以外，检测节点无法获知流 f 中其他字段内容。上述结论的严格证明详见文献[9]。

在 2.2 节攻击者模型中，设定检测节点为半诚实模型，即检测节点严格执行协议步骤，但试图窃取用户隐私信息。为此，检测节点可采取一种隐蔽攻击方式，将试图获取的信息作为入侵检测特征同加密流量内容进行比对。如检测节点可将“简历投递”“薪酬期望”等作为入侵检测特征。若在加密流量中匹配出此类关键词，检测节点在无法获知流量其他内容的前提下，也能推断出该流量的主要信息，威胁用户个人隐私。为抵御此类攻击，如 2.1 节中所述，用户终端在解密流量内容后，将判断流量内容中是否涉及用户敏感信息。通过用户终端的主动检查，能够发现检测节点的上述攻击行为，从而使此类攻击失效。

2) 恶意用户终端成功通过验证的概率分析

为解决恶意用户终端的任意输入问题，在 3.3 节中提出随机验证策略。随机选取 e 个流量内容片段，计算对应 $\widetilde{T_c^i}$ 的值，并同用户终端 u_t 发送的 T_c^i 值进行比较。若存在不一致，则表明用户终端 u_t 在执行 PSI 协议时存在输入替代，如将可能的恶意特征替换为正

常字符，或直接使用随机数据替代流量内容等。由于为随机验证，恶意用户终端能以一定的概率通过输入验证。具体的概率 Pr 计算如下。

情形 1 u_z 为恶意用户终端。设集合 U 中恶意用户终端总数为 m, $m<|C|$。如 2.2 节攻击者模型中所设定，恶意用户终端可以相互合谋以完成特定攻击目标。此时，u_z 不需要计算 $\widetilde{T_c^i}$ 的值，可直接发送对应的 T_c^i 值至检测节点。检测节点通过比对，无法发现 u_z 实施了输入替代，u_z 成功通过输入验证。由于 u_z 为随机选择，情形 1 发生的概率为

$$\mathrm{Pr}_1=\frac{m}{|U|} \tag{19}$$

情形 2 u_z 为正常用户终端。假定恶意用户终端 u_z 对流量内容中 r ($r\geqslant 1$)个连续字符进行了修改。在 3.1 节用户终端预处理步骤中，修改的 r 个字符将平均出现 $r+l-1$ 次。在随机选择 e 个字符段时，选择的字符段均不包含 r 个修改字符的概率为 $\frac{\mathrm{C}_{|C|-(r+l-1)}^{e}}{\mathrm{C}_{|C|}^{e}}$。此时，恶意用户终端成功通过输入验证的概率为

$$\mathrm{Pr}_2=\left(1-\frac{m}{|U|}\right)\frac{\mathrm{C}_{|C|-(r+l-1)}^{e}}{\mathrm{C}_{|C|}^{e}} \tag{20}$$

当 $e=|C|$时，$\mathrm{Pr}_2=0$。最终的 Pr 为

$$\mathrm{Pr}=\mathrm{Pr}_1+\mathrm{Pr}_2=\frac{m}{|U|}+\left(1-\frac{m}{|U|}\right)\frac{\mathrm{C}_{|C|-(r+l-1)}^{e}}{\mathrm{C}_{|C|}^{e}}=$$

$$\frac{m}{|U|}+\left(1-\frac{m}{|U|}\right)\frac{\prod_{i=1}^{r}\left(|C|-e-r-l+1+i\right)}{\prod_{i=1}^{r}\left(|C|-r-l+1+i\right)} \tag{21}$$

若恶意用户终端通过调换 r 个字符段的位置实现输入代替，Pr 的计算式依然为式(21)。分析式(21)，当 $\frac{\mathrm{C}_{|C|-(r+l-1)}^{e}}{\mathrm{C}_{|C|}^{e}}=z$ 值固定时，$\mathrm{Pr}=(1-z)\frac{m}{|U|}+z$，Pr 值随着 $\frac{m}{|U|}$ 的值，即恶意用户终端所占比例递增而递增；当 $\frac{m}{|U|}=z$ 值固定时，$\mathrm{Pr}=(1-z)\cdot\frac{\prod_{i=1}^{r}\left(|C|-e-r-l+1+i\right)}{\prod_{i=1}^{r}\left(|C|-r-l+1+i\right)}+z$，随着 r 或 e 或 l 值递增，Pr 值递减。具体参数值的选取和验证将于第 5 节中阐述。值得注意的是，式(21)为在随机选择一个用户终端作为验证方时，恶意用户终端通过输入验证的概率值。若随机选择 n（$n\geqslant 1$）个用户终端作为验证方，则 $\mathrm{Pr}=\left[\frac{m}{|U|}+\left(1-\frac{m}{|U|}\right)\frac{\mathrm{C}_{|C|-(r+l-1)}^{e}}{\mathrm{C}_{|C|}^{e}}\right]^n$。因而可通过增加 n 的值使 Pr 快速趋向于 0。

4.2 性能分析

由于检测节点一般由专用服务器实现，且检测节点的性能分析过程与用户终端类似，本节重点分析用户终端的运行性能。在方法的运行过程中，用户终端需要进行数据预处理以及计算式(5)～式(8)。其中，数据预处理仅需遍历加密网络流量 f 的内容一次、式(7)中的零知识证明只需要计算 Hash 一次，产生的性能消耗极低。因此，对用户终端性能产生主要影响的是式(6)和式(8)中的模幂运算和 Hash 计算。令模幂运算的计算资源开销（如 CPU 执行时间）为 λ_1，Hash 函数的计算资源开销为 λ_2，字符串关键词对应的集合 S_1 的大小为 $|S_1|$，正则表达式关键词对应的集合 S_2 的大小为 $|S_2|$，则用户终端所需的计算资源开销 λ_c 为

$$\lambda_c\propto\left(|C|+|S_1|+|S_2|\right)\lambda_1+2|C|\lambda_2 \tag{22}$$

由式(22)可知，用户终端所需的计算资源开销与集合 C 和集合 S 的大小呈线性相关。

在计算过程中，用户终端需遍历集合$\{M_j\}$和集合 C 一次，计算和存储结果。集合 C 的大小为 $|C|$，集合$\{M_j\}$的大小为 $|S_1|+|S_2|$，计算结果所占比特数与模数 p 长度相同（如 512 bit）。因此，用户终端的内存消耗 λ_m 可表示为

$$\lambda_m\propto\left(|C|+|S_1|+|S_2|\right)|p| \tag{23}$$

同样地，存储资源消耗与集合 C 和集合 S 的大小呈线性相关。

用户终端的网络通信带宽消耗分析如下。如 3.1 节中所述，检测节点需将集合$\{M_j\}$发送至用户终端，用户终端需将集合$\{M_j^{'}\}$和$\{T_c^i\}$发送至检测节点。因而消耗的网络带宽正比于集合中元素个数和字符段长度 l，如式(24)所示。

$$\lambda_n\propto\left(|C|+|S_1|+|S_2|\right)l \tag{24}$$

设未分割前的流量内容字符串长度为 L_f，入侵

检测特征关键词字符串长度为 L_I，字符段的长度为 l，入侵检测特征关键词总数量为 h，则有

$$|C| = L_f - l + 1 \tag{25}$$

$$|S_1| + |S_2| = h(L_I - l + 1) \tag{26}$$

将式(25)、式(26)分别代入式(22)～式(24)，可得

$$\lambda_c \propto (L_f + hL_I + h + 1)\lambda_1 + 2(L_f + 1)\lambda_2 - [(h+1)\lambda_1 + 2\lambda_2]l \tag{27}$$

$$\lambda_m \propto (L_f + hL_I + h + 1)|p| - (h+1)|p|l \tag{28}$$

$$\lambda_n \propto (L_f + hL_I + h + 1)l - (h+1)l^2 \tag{29}$$

由式(27)和式(28)知，在总数据量（L_f、L_I 以及 h）一定的情形下，随着 l 值递增，用户终端的计算和存储资源开销递减。以 l 为自变量，对式(29)求一阶导数，可得当 $l = \dfrac{L_f + hL_I + h + 1}{2(h+1)}$ 时，所消耗的网络带宽最大。因此，当 $l \leqslant \dfrac{L_f + hL_I + h + 1}{2(h+1)}$ 时，随着 l 值的递增，用户终端的计算和存储资源开销递减，但消耗的网络带宽递增；当 $l > \dfrac{L_f + hL_I + h + 1}{2(h+1)}$ 时，随着 l 值的递增，用户终端的计算和存储资源开销递减，消耗的网络带宽也递减。考虑到 $l \leqslant L_I$ 这个约束条件，在实际应用中，可通过设置较长的入侵检测关键词来减少用户终端的资源消耗。具体实验验证见 5.2 节。

5　实验验证

本节主要验证方法对恶意加密流量检测确认的准确性和对用户终端的性能影响程度。实验中运行 Windows XP 虚拟机作为用户终端，检测节点为 Dell PowerEdge R730 服务器。虚拟机配置为运行单个 CPU，内存为 2 GB，模拟较低性能终端设备。检测节点服务器与用户终端位于同一局域网，并配备路由转发功能。检测节点能够捕获用户终端所产生的所有流量。

在用户终端中，正常应用（如浏览器的密钥提取）通过配置 SSLKEYLOGFILE 系统变量实现。对于恶意软件，其加密网络流量的密钥提取通过 mitmproxy 实现，提取的密钥存储同样存于 SSLKEYLOGFILE 文件中。加密流量的解密和十六进制内容导出由 tshark 工具完成。入侵检测规则为开源的 ET Suricata 规则，包括 attack_response、malware、shellcode、Web 等规则文件。入侵检测规则和网络流量内容的分割由 Python 代码完成。机器学习算法选定为随机森林算法，分类特征包括 TLS 证书、报文长度分布等。

PSI 代码由 Python 代码实现。其中，在用户终端处运行 Flask Python Web 框架，监听 5000 端口。用户终端同检测节点采用 HTTP 进行数据的接收和发送。在数据处理时，流量内容和关键词首先以十六进制字符形式存于文件，PSI 代码从文件中读取字符段并转换为对应数字进行模幂运算和 Hash 计算。Hash 计算利用 SHA256 完成，模幂运算则由高精度运算库 gmpy2 实现。此外，在 PSI 代码中额外增加 CPU 执行时间（利用 perf_counter 函数实现）和内存消耗（利用 psutil 库实现）统计功能，便于计算方法的系统资源开销。鉴于 TLS 已成为互联网安全传输的事实标准[35]，实验中主要针对 TLS 加密流量进行测试。

5.1　功能验证

功能验证主要包含 3 个方面的内容：1) 验证对恶意加密流量检测的误报消除效果，由减少的误报数量衡量；2) 验证对检测召回率（Recall）的影响，由漏报数量衡量；3) 恶意用户终端成功通过输入验证的可能性，由 Pr 衡量。实际上，恶意加密流量的检测召回率主要由机器学习算法决定，但对检测结果进行进一步确认可能增加漏报，从而降低召回率。实验中，设定 l 的值，即字符段长度为 5。这是由于选用的 ET Suricata 规则中最短的关键词长度为 5。

为验证对误报的消除效果，在虚拟机中使用 Firefox 浏览器访问 Alexa Top 100 网站。依据文献[36]，可认为 Alexa Top 100 网站为合法网站，产生的流量为正常流量。实验中使用 Shell 编程和 iMacros 插件实现 Firefox 浏览器的自动访问和自动链接点击，对每个网址的访问时间设定为 1 min，共捕获 15 887 条 TLS 流量。实验结果如表 2 所示。检测端使用随机森林算法进行判断，检测出 13 条 TLS 流量为恶意流量。经过特征安全匹配后，有 12 条 TLS 流量被确认为正常流量，即消除误报流量 12 条，误报数量减少了 92.3%。对最终的误报流量进行人工分析发现，流量中含有“/perl”字段，从而触发了入侵检测规则。

表 2 误报消除实验结果

指标	数值
捕获的 TLS 总数量/条	15 887
判断为恶意的 TLS 流量数量/条	13
确认为误报的 TLS 流量数量/条	12

为验证对恶意加密流量检测召回率的影响，从互联网中下载 EXE 类型病毒样本 1 047 例。将下载的病毒样本逐一运行于虚拟机中，并使用 tshark 软件进行网络流量捕获。为避免干扰，完成一例病毒样本测试后，将虚拟机还原至初始状态，然后运行另一例病毒样本。实验结果如表 3 所示。由于部分样本无法正常运行，实验中共捕获 278 条恶意 TLS 流量。在检测端运行随机森林算法，共检测出 269 条。但如表 3 所示，经过特征安全匹配后，有 2 条恶意 TLS 流量被判断为正常流量，从而产生了漏报。对这 2 条恶意 TLS 流量进行人工分析发现，其数据内容为压缩或可执行文件，从而导致规则匹配失效。

表 3 检测漏报实验结果

指标	数值
捕获的 TLS 总数量/条	278
判断为恶意的 TLS 流量数量/条	269
确认为恶意的 TLS 流量数量/条	267
确认所产生的漏报数量/条	2

综合表 2 和表 3 的实验结果可知，本文提出的方法能够有效减少误报，但同时也增加了一些漏报数量。误报和漏报产生的原因是规则的精确度有限，如将“/perl”字段判断为恶意攻击、无法匹配压缩和可执行文件内容等。因此，在实际使用中，可采用或制定更精确的入侵检测特征，提升方法的准确性和实用性。为验证上述思路，购买了商用版 Snort Ruleset 对表 2 中判断为恶意流量的 13 条正常 TLS 流量和表 3 中检测出的 269 条恶意 TLS 流量进行再次确认。实验结果表明，最新规则能够排除所有误报，且能确认所有恶意 TLS 流量。

如 4.1 节中所述，恶意用户终端能以一定概率通过输入验证。为验证式(21)的正确性，首先随机产生长度为 1 000 的随机十六进制字符串，并随机选择 r 个连续字符进行修改。对修改后的字符串进行滑动分割，分割长度为 l，形成集合 C。接着，产生 0～1 的随机数 R。若 $R \leqslant \frac{m}{|U|}$（$\frac{m}{|U|}$ 的值为设定的实验参数），则直接判定恶意用户通过输入验证。否则，从集合 C 中随机选择 e 个字符段，若选出的 e 个字符段均不包含修改后的字符，则直接判定恶意用户通过输入验证。最后，重复实验 10 000 次，统计通过验证的次数并除以 10 000，即实验求出的 Pr 的值。不同参数条件下的实验结果如图 3～图 6 所示。

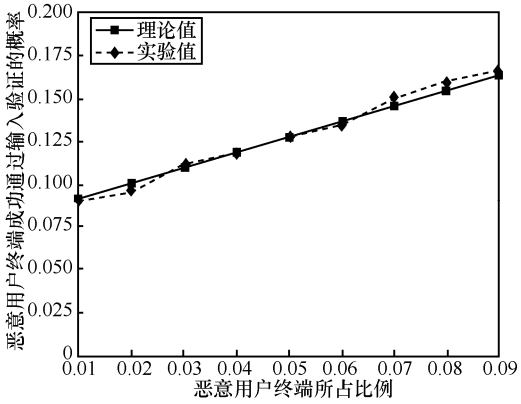


图 3 设定 r=3, l=10, e=300 时，随恶意用户终端所占比例不同的取值所对应的 Pr 值

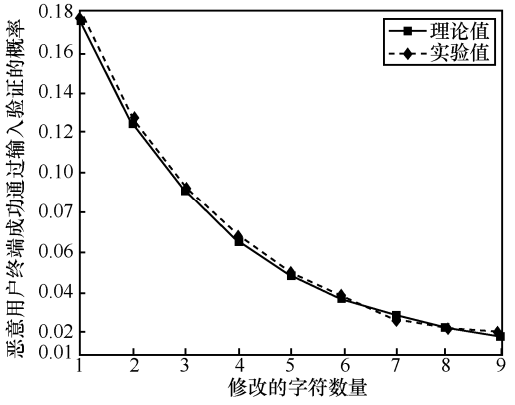


图 4 设定 $\frac{m}{|U|}$=0.01, l=10, e=300 时，修改的字符数量不同的取值所对应的 Pr 值

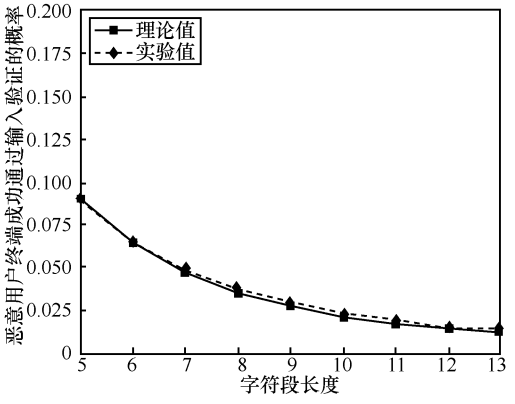


图 5 设定 $\frac{m}{|U|}$=0.01, r=3, e=300 时，字符段长度不同的取值所对应的 Pr 值

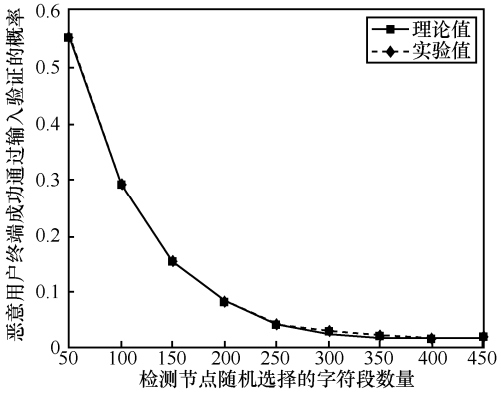


图 6　设定 $\frac{m}{|U|}$=0.01, r=3, l=10 时，字符段数量不同的取值所对应的 Pr 值

在图 3～图 6 中，理论值为式(21)的计算结果，理论值与实验值基本一致。实验结果表明，随着恶意用户终端所占比例的递增，恶意用户终端成功通过输入验证的概率 Pr 递增；随着修改的字符数量、字符串长度以及检测节点随机选择的字符段数量的递增，Pr 值递减。特别地，当随机选择的字符段数量超过总字符段数量的一半时，Pr 值趋近于 0。在实际应用中，可选择较大的 l 和 e 值，减少恶意用户终端成功通过输入验证的可能性。实验中还验证了其他多组参数，如 r=4, l=11, e=200 和 $\frac{m}{|U|}$=0.02, l=10, e=400 等。实验结果均与理论计算结果一致，因此不再叙述。

5.2　性能验证

本节给出不同参数条件下，用户终端的资源消耗程度。具体地，在总数据量一定的前提下，通过设置不同的字符串长度，统计用户终端的计算性能开销、内存占用量和网络带宽消耗。在实验中，计算性能开销由 CPU 执行时间来衡量，内存占用量由使用的内存字节数表示，网络带宽消耗由发送和接收的数据分组字节数表示。

首先，验证式(27)和式(28)的正确性。在式(27)中，区分模幂运算和 Hash 计算的代价为 λ_1 和 λ_2。在实验中，利用 CPU 执行时间统一衡量模幂运算和 Hash 计算的所需代价。为此，在隐私保护集合交集代码中使用 perf_counter 函数统计式(4)～式(8)所需的计算时间。设定网络流内容共有 10 000 个字符，入侵检测特征长度为 20，入侵检测特征数量为 1 000，字符段长度为 1～9。对于相同数据，CPU 执行时间统计结果会略有差别。因此，对于同一参数运行 10 次，取 CPU 执行时间的平均值为最后实验结果，如图 7 所示。实验结果表明，随着字符段长度的递增，CPU 执行时间递减，同式(27)的分析结果一致。

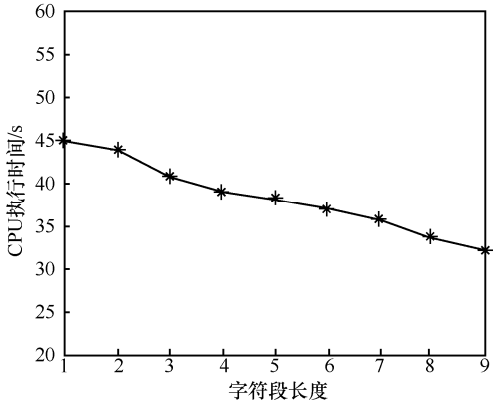


图 7　不同字符段长度所对应的 CPU 执行时间

程序所占用的内存字节由 psutil 库统计。同样地，设定网络流内容共有 10 000 个字符，入侵检测特征长度为 20，入侵检测特征数量为 1 000。为反映内存占用的变化趋势，字符段长度 l 值设定为 1、4、7、10、13、16 和 19。对于同一参数，运行 10 次，取内存占用的平均值为最后实验结果，如图 8 所示。实验结果表明，随着字符段长度的递增，所占用的内存大小递减，与式(28)的分析结果一致。

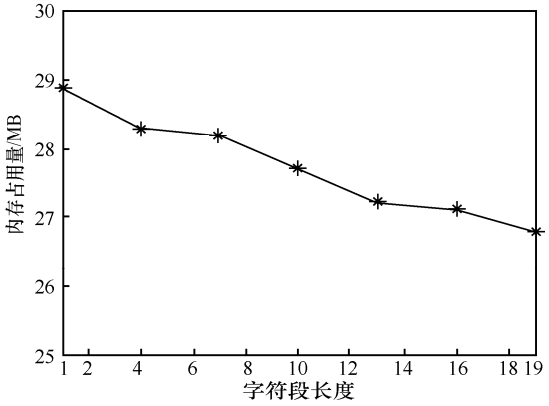


图 8　不同字符段长度所对应的内存占用量

其次，验证式(29)的正确性。式(29)表明，在总数量一定的前提下，当 $l \leqslant \frac{L_f + hL_I + h + 1}{2(h+1)}$ 时，随着 l 值的递增，网络带宽消耗递增；当 $l > \frac{L_f + hL_I + h + 1}{2(h+1)}$ 时，随着 l 值的递增，网络带宽消耗递减。设定网络流内容共有 10 000 个字符，入侵检测特征长度为 20，入侵检测特征数量为 1 000。

依据上述分析，当 l=15 时，网络带宽消耗最多。实验中调节 l 的值从 12 递增至 19，在用户终端处使用 tshark 工具捕获端口 5000 的网络流量，并统计流中数据字节数（不包括报文头部字段）。不同字符段长度所对应的网络带宽消耗如图 9 所示。实验结果同理论分析相一致。实验中还测试了其他不同参数，结果趋势均一致，故不再赘述。

综上所述，当 $l \geqslant 15$ 时，随着 l 值的增大，CPU 执行时间、内存占用量、网络带宽消耗以及恶意用户终端成功通过输入验证的概率均呈下降趋势。因此在实际应用中可设置较长的入侵检测特征关键词，并通过增加 l 的值以提升方法性能。

最后，给出实际运行时的资源消耗统计。如 5.1 节中所述，通过访问 Alexa Top 100 网站和实际运行 1 047 例恶意样本，共捕获 16 165 条 TLS 流量。其中有 282（即 13+269）条 TLS 流量被随机森林算法判别为恶意流量。在对这 282 条 TLS 流量进行实际确认时，分别统计用户终端的 CPU 执行时间、CPU 占用率、网络带宽消耗以及内存占用量，并对统计结果取均值作为最后数值，如表 4 所示。

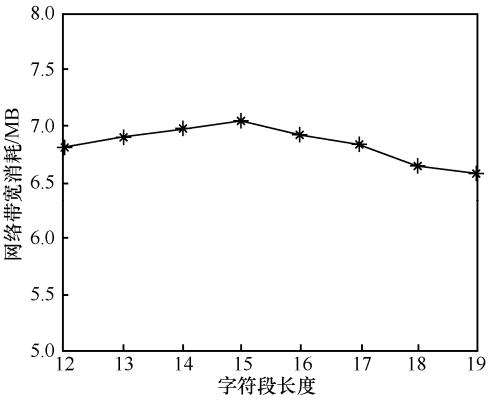


图 9 不同字符段长度所对应的网络带宽消耗

表 4 实际确认时用户终端的性能消耗

指标（均值）	数值
CPU 执行时间/s	58.3
CPU 占用率	24.9%
网络带宽消耗/MB	11.6
内存占用量/MB	52.1

实验结果表明，在执行确认过程中，用户终端需要进行 1 min 左右的频繁计算（约 25%的 CPU 使用率），平均占用 52.1 MB 的内存，消耗的网络带宽平均为 11.6 MB，可适用于普通 PC 和智能手机。需要注意的是，对于正常用户终端，其仅在检测节点产生误报时才参与检测确认过程。若误报率为 0.01%，正常用户终端平均产生 10 000 条加密流量才参与一次确认。在其他情形时，检测节点与用户终端间无网络交互，用户终端也不需要进行模幂和 Hash 计算，本文提出的方法的运行仅需极少的系统资源。因此本文提出的恶意加密流量检测确认方法对用户终端系统的实际运行影响较小。

将本文提出的方法同目前实用的基于密文解密的恶意加密流量检测方法[11,13]进行对比。本文提出的方法只需对可疑的加密网络流量进行解密，而文献[11,13]中的方法需对所有加密网络流量进行解密。本文提出的方法运用 PSI 协议进行特征匹配，除求出的交集外，检测节点无法获知网络流量中其他内容，而文献[11,13]中检测节点能够获得用户终端的所有网络流量内容。因此，本文提出的方法具有更好的隐私保护特征，且如表 4 所示，本文提出的方法运行仅需极少的系统资源。本文提出的方法可以与基于机器学习的恶意加密流量检测方法相配合，以一种新方式实现恶意加密流量的安全、有效监管。

6 结束语

本文提出并实现了一种支持隐私保护的恶意加密流量检测确认方法。在提出的方法中，可以使用任意机器学习算法进行恶意加密流量的检测判断，并支持字符串关键词、正则表达式关键词以及关键词位置信息判断，因而具有良好的适用性和可扩展性。在确认过程中，用户终端无法获得检测节点的入侵检测特征内容，检测节点无法获知交集以外的用户终端其他通信内容，从而实现了双方的数据隐私保护。利用真实加密网络流量进行测试，实验结果表明，本文提出的方法能减少 92.3%的误报。在进行检测确认时，CPU 占用率接近 25%，但此过程持续时间短，平均为 58.3 s，且发生的频率低，因而对用户终端的系统性能影响有限。未来工作包括将用户终端功能同 ClamWin 等开源杀毒软件相结合，优化程序实现提升系统性能，并在校园网等实际网络中部署使用。

参考文献：

[1] 罗军舟, 何源, 张兰, 等. 云端融合的工业互联网体系结构及关键技术[J]. 中国科学: 信息科学, 2020, 50(2): 195-220.

LUO J Z, HE Y, ZHANG L, et al. The architecture and key technologies for an industrial Internet with synergy between the cloud and clients[J]. Scientia Sinica (Informationis), 2020, 50(2): 195-220.

[2] DING D R, HAN Q L, XIANG Y, et al. A survey on security control and attack detection for industrial cyber-physical systems[J]. Neurocomputing, 2018, 275: 1674-1683.

[3] 张蕾, 崔勇, 刘静, 等. 机器学习在网络空间安全研究中的应用[J]. 计算机学报, 2018, 41(9): 1943-1975.
ZHANG L, CUI Y, LIU J, et al. Application of machine learning in cyberspace security research[J]. Chinese Journal of Computers, 2018, 41(9): 1943-1975.

[4] ANDERSON B, PAUL S, MCGREW D. Deciphering malware's use of TLS (without decryption)[J]. Journal of Computer Virology and Hacking Techniques, 2018, 14(3): 195-211.

[5] WANG W, ZHU M, ZENG X W, et al. Malware traffic classification using convolutional neural network for representation learning[C]//Proceedings of 2017 International Conference on Information Networking (ICOIN). Piscataway: IEEE Press, 2017: 712-717.

[6] HAN D Q, WANG Z L, CHEN W Q, et al. DeepAID: interpreting and improving deep learning-based anomaly detection in security applications[C]//Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2021: 3197-3217.

[7] FROLOV S, WUSTROW E. The use of TLS in censorship circumvention[C]//Proceedings of 2019 Network and Distributed System Security Symposium. Reston: Internet Society, 2019: 1-15.

[8] HO C Y, LAI Y C, CHEN I W, et al. Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems[J]. IEEE Communications Magazine, 2012, 50(3): 146-154.

[9] DE CRISTOFARO E, KIM J, TSUDIK G. Linear-complexity private set intersection protocols secure in malicious model[C]//2010 International Conference on the Theory and Application of Cryptology and Information Security. Berlin: Springer, 2010: 213-231.

[10] ZHAO C, ZHAO S N, ZHAO M H, et al. Secure multi-party computation: theory, practice and applications[J]. Information Sciences, 2019, 476: 357-372.

[11] CARNAVALET X D, MANNAN M. Killed by proxy: analyzing client-end TLS interception software[C]//Proceedings of 2016 Network and Distributed System Security Symposium. Reston: Internet Society, 2016: 1-17.

[12] O'NEILL M, RUOTI S, SEAMONS K, et al. TLS proxies: friend or foe? [C]//Proceedings of the 2016 Internet Measurement Conference. New York: ACM Press, 2016: 551-557.

[13] NAYLOR D, SCHOMP K, VARVELLO M, et al. Multi-context TLS (mcTLS)[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 199-212.

[14] LIU C, CUI Y, TAN K, et al. Building generic scalable middlebox services over encrypted protocols[C]//Proceedings of 2018 IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2018: 2195-2203.

[15] SHERRY J, LAN C, POPA R A, et al. BlindBox: deep packet inspection over encrypted traffic[C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. New York: ACM Press, 2015: 213-226.

[16] NING J T, POH G S, LOH J C, et al. PrivDPI: privacy-preserving encrypted traffic inspection with reusable obfuscated rules[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2019: 1657-1670.

[17] LAI S Q, YUAN X L, SUN S F, et al. Practical encrypted network traffic pattern matching for secure middleboxes[J]. IEEE Transactions on Dependable and Secure Computing, 2021, PP(99): 1.

[18] IOVINO V, PERSIANO G. Hidden-vector encryption with groups of prime order[C]//2008 International Conference on Pairing-Based Cryptography. Berlin: Springer, 2008: 75-88.

[19] ANDERSON B, CHI A, DUNLOP S, et al. Limitless HTTP in an HTTPS world: inferring the semantics of the HTTPS protocol without decryption[C]//Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy. New York: ACM Press, 2019: 267-278.

[20] HELLEMONS L, HENDRIKS L, HOFSTEDE R, et al. SSHCure: a flow-based SSH intrusion detection system[C]//2012 IFIP International Conference on Autonomous Infrastructure, Management and Security. Berlin: Springer, 2012: 86-97.

[21] HE G F, ZHANG T, MA Y Y, et al. A novel method to detect encrypted data exfiltration[C]//Proceedings of 2014 Second International Conference on Advanced Cloud and Big Data. Piscataway: IEEE Press, 2014: 240-246.

[22] HE G F, XU B F, ZHANG L, et al. On-device detection of repackaged android malware via traffic clustering[J]. Security and Communication Networks, 2020, 2020: 8630748.

[23] CHEN Y C, LI Y J, TSENG A, et al. Deep learning for malicious flow detection[C]//Proceedings of 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications. Piscataway: IEEE Press, 2017: 1-7.

[24] 翟明芳, 张兴明, 赵博. 基于深度学习的加密恶意流量检测研究[J]. 网络与信息安全学报, 2020, 6(3): 66-77.
ZHAI M F, ZHANG X M, ZHAO B. Survey of encrypted malicious traffic detection based on deep learning[J]. Chinese Journal of Network and Information Security, 2020, 6(3): 66-77.

[25] 何高峰, 司勇瑞, 徐丙凤. 针对 Android 移动应用的恶意加密流量标注方法研究[J]. 计算机工程, 2020, 46(7): 116-121, 128.
HE G F, SI Y R, XU B F. Research on malicious encrypted traffic annotation method for android mobile application[J]. Computer Engineering, 2020, 46(7): 116-121, 128.

[26] JAN S T K, HAO Q Y, HU T R, et al. Throwing darts in the dark? detecting bots with limited data using neural data augmentation[C]//Proceedings of 2020 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2020: 1190-1206.

[27] LAN C, SHERRY J, POPA R A, et al. Embark: securely outsourcing middleboxes to the cloud[C]//2016 USENIX Symposium on Net-

worked Systems Design and Implementation. Berkeley: USENIX Association, 2016: 255-273.

[28] GUO Y, WANG M Y, WANG C, et al. Privacy-preserving packet header checking over in-the-cloud middleboxes[J]. IEEE Internet of Things Journal, 2020, 7(6): 5359-5370.

[29] CASTELLUCCIA C, CRISTOFARO E D, PERITO D. Private information disclosure from web searches[C]//2010 International Symposium on Privacy Enhancing Technologies Symposium. Berlin: Springer, 2010: 38-55.

[30] SEOK J, CHOI M, KIM J, et al. A comparative study on performance of open source IDS/IPS snort and suricata[J]. Journal of the Korea Society of Digital Industry and Information Management, 2016, 12(1): 89-95.

[31] CHAUM D. Zero-knowledge undeniable signatures[C]//1990 Workshop on the Theory and Application of Cryptographic Techniques. Berlin: Springer, 1990: 458-464.

[32] COHEN H, PORAT E. Fast set intersection and two-patterns matching[J]. Theoretical Computer Science, 2010, 411(40/41/42): 3795-3800.

[33] LI N. Research on Diffie-Hellman key exchange protocol[C]//Proceedings of 2010 2nd International Conference on Computer Engineering and Technology. Piscataway: IEEE Press, 2010: 634-637.

[34] DIEM C. On the discrete logarithm problem in elliptic curves[J]. Compositio Mathematica, 2011, 147(1): 75-104.

[35] 潘吴斌, 程光, 郭晓军, 等. 网络加密流量识别研究综述及展望[J]. 通信学报, 2016, 37(9): 154-167.
PAN W B, CHENG G, GUO X J, et al. Review and perspective on encrypted traffic identification research[J]. Journal on Communications, 2016, 37(9): 154-167.

[36] ZENG F, CHANG S, WU X C. Classification for DGA-based malicious domain names with deep learning architectures[J]. International Journal of Intelligent Information Systems, 2017, 6(6): 67-71.

[作者简介]

何高峰（1984− ），男，安徽安庆人，博士，南京邮电大学讲师、硕士生导师，主要研究方向为网络异常检测、可证明安全的网络安全防御等。

魏千峰（1997− ），男，江苏徐州人，南京邮电大学硕士生，主要研究方向为网络流量异常检测。

肖咸财（1998− ），男，江西赣州人，南京邮电大学硕士生，主要研究方向为加密网络流量分析。

朱海婷（1983− ），女，江苏如皋人，博士，南京邮电大学讲师、硕士生导师，主要研究方向为网络管理和网络性能优化。

徐丙凤（1986− ），女，安徽安庆人，博士，南京林业大学讲师、硕士生导师，主要研究方向为网络安全威胁建模与评估。