一种基于HTTP/3传输特性的加密视频识别方法

吴 桦[1),2),3)] 倪珊珊[1)] 罗 浩[1)] 程 光[1),2),3),4)] 胡晓艳[1),2),3)]

[1)](东南大学 网络空间安全学院 南京 211189)

[2)](网络通信与安全紫金山实验室 南京 211111)

[3)](网络空间国际治理研究基地(东南大学) 南京 211189)

[4)](江苏省泛在网络安全工程研究中心 南京 211189)

摘 要 视频流量逐渐在网络中占据主导地位,且视频平台大多对其进行加密传输.虽然加密传输视频可以有效保护用户隐私,但是也增加了监管有害视频传播的难度.现有的加密视频识别方法基于TCP(Transmission Control Protocol)传输协议头部信息和HTTP/1.1(Hypertext Transfer Protocol Version 1.1)的传输模式,提取应用层音视频数据单元传输长度序列来实现视频识别.但是随着基于UDP(User Datagram Protocol)的QUIC(Quick UDP Internet Connections)协议及基于QUIC实现的HTTP/3(Hypertext Transfer Protocol Version 3)协议应用于视频传输,已有方法不再适用.HTTP/3协议缺少类似TCP的头部信息,且使用了多路复用机制,并对几乎所有数据进行了加密,此外,视频平台开始使用多片段合并分发技术,这给从网络流量中精准识别加密视频带来了巨大挑战.本文基于HTTP/3协议中的控制信息特征,提出了从HTTP/3加密视频流中提取数据传输特征并进行修正的方法,最大程度复原出应用层音视频长度特征.面向多片段合并分发导致的海量匹配问题,本文基于明文指纹库设计了键值数据库来实现视频的快速识别.实验结果表明,本文提出的基于HTTP/3传输特性的加密视频识别方法能够在包含36万个真实视频指纹的YouTube大规模指纹库中达到接近99%的准确率、100%的精确率以及99.32%的F1得分,对传输过程中加入了填充帧的Facebook平台,在包含28万个真实视频指纹的大规模指纹库中达到95%的准确率、100%的精确率以及96.45%的F1得分,在具有同样特性的Instagram平台中,最高可达到97.57%的F1得分,且本方法在所有指纹库中的平均视频识别时间均低于0.4秒.本文的方法首次解决了使用HTTP/3传输的加密视频在大规模指纹库场景中的识别问题,具有很强的实用性和通用性.

关键词 HTTP/3协议;QUIC协议;基于HTTP的动态自适应流媒体;视频识别;加密流量

中图法分类号 TP393 **DOI号** 10.11897/SP.J.1016.2024.01640

An Encrypted Video Recognition Method Based on the Transmission Characteristics of HTTP/3

WU Hua[1),2),3)] NI Shan-Shan[1)] LUO Hao[1)] CHENG Guang[1),2),3),4)] HU Xiao-Yan[1),2),3)]

[1)](*School of Cyber Science and Engineering, Southeast University, Nanjing* 211189)

[2)](*Purple Mountain Laboratories for Network and Communication Security, Nanjing* 211111)

[3)](*International Governance Research Base of Cyberspace (Southeast University), Nanjing* 211189)

[4)](*Jiangsu Province Engineering Research Center of Security for Ubiquitous Network, Nanjing* 211189)

Abstract The proliferation of encrypted video streaming has significantly impacted internet traffic, with most platforms opting for encryption to safeguard user privacy. However, this

收稿日期:2023-08-14;在线发布日期:2024-04-23.本课题得到国家重点研发项目(No. 2021YFB3101403)资助. **吴 桦**(通信作者),博士,副教授,中国计算机学会(CCF)会员,主要研究领域为加密流量分析、网络空间安全、网络态势感知. E-mail: hwu@seu.edu.cn. **倪珊珊**,硕士研究生,主要研究领域为加密流量分析、加密视频识别. E-mail: ssni@seu.edu.cn. **罗 浩**,硕士研究生,主要研究领域为加密流量分析、加密视频识别. E-mail: hluo@seu.edu.cn. **程 光**,博士,教授,中国计算机学会(CCF)会员,主要研究领域为网络空间安全监测与防护、网络流量大数据分析、僵尸网络和APT攻击检测. E-mail:chengguang@seu.edu.cn. **胡晓艳**,博士,副教授,中国计算机学会(CCF)会员,主要研究领域为加密流量分析、网络空间安全、未来网络体系结构. E-mail: xyhu@seu.edu.cn.

approach also poses substantial challenges to content monitoring and the regulation of potentially harmful broadcasts. Traditional techniques for the identification of encrypted video streams typically utilize TCP (Transmission Control Protocol) header information and the transmission mode of HTTP/1.1 (Hypertext Transfer Protocol Version 1.1), extracting sequences of application-layer audio and video data unit lengths to recognize video content. However, the evolution of Internet protocols, notably with the adoption of the UDP (User Datagram Protocol)-based QUIC (Quick UDP Internet Connections) protocol and its subsequent implementation in HTTP/3 (Hypertext Transfer Protocol Version 3), has made these traditional methods inadequate. HTTP/3 significantly differs from its predecessors by lacking similar header information as TCP, using multiplexing to handle multiple data streams simultaneously, and encrypting nearly all transmitted data. These changes, along with the adoption of segmented and merged distribution strategies by video platforms, complicate the accurate identification of encrypted video content from network traffic. The necessity for new methods to tackle these challenges led to the development of the approach detailed in this paper, which capitalizes on specific control information characteristics within the HTTP/3 protocol. Our method focuses on extracting and refining data transmission features from HTTP/3 encrypted video streams, enabling the reconstruction of application-layer audio and video length characteristics. This reconstruction is crucial for identifying video content accurately. To handle the challenges posed by segmented merging distribution, we developed a key-value database that utilizes a plaintext fingerprint library. This database design facilitates rapid video identification by quickly matching incoming encrypted video data against stored fingerprints. Experimental validation of our method was conducted using a comprehensive dataset comprising 362, 502 real video fingerprints from YouTube. The results show that our method achieves nearly 99% accuracy, 100% precision, and 99.32% F1 score. These metrics attest to the robustness and reliability of our technique. Further experiments on Facebook, where padding frames are commonly introduced during transmission, demonstrated commendable performance within a large-scale fingerprint database containing 283, 895 real video fingerprints. Specifically, our method achieved a notable accuracy of 95%, a precision of 100%, and an F1 score of 96.45%. On the Instagram platform, which shares the same features as Facebook, our method can achieve a maximum F1 score of 97.57%. One of the standout features of our approach is its efficiency, with the average video identification time across all fingerprint databases being less than 0.4 seconds. This speed is vital for applications that require real-time decision-making, such as automated content moderation systems. In conclusion, this paper presents a pioneering method for the identification of encrypted videos transmitted via HTTP/3, filling the existing gap in encrypted video content recognition. The approach not only enhances the ability to monitor and regulate encrypted video content but also respects user privacy. This balance is critical in today's digital age, where privacy concerns are paramount. Our method can simultaneously address issues related to internet pollution regulation and citizen privacy protection without altering existing network architectures, thereby possessing significant practical value.

Keywords Hypertext Transfer Protocol Version 3; Quick UDP Internet connections; dynamic adaptive streaming over HTTP; video recognition; encrypted traffic

1 引 言

随着通信技术的进步和移动互联网的发展,网络交互场景越来越丰富,网络应用种类越来越多,互联网已经成为社会生活的一部分.通过流量分析对网络空间进行感知是保障国家安全的重要方法.

互联网的高速演化导致了网络流量的加密化和协议的复杂化,这给网络流量分析带来了挑战.目前已有较多的研究展开了对加密流量的分析.根据研究目标的不同,主要可以分为流量分类、流量细粒度信息识别以及对流量分析的支撑研究.本文的研究属于流量细粒度信息识别,是对视频流量中承载的内容进行识别.

视频网络流量在互联网中的比重不断增加.互联网流量检测机构 Sandvine 的 2023 年全球互联网现象报告显示①,视频流量在 2022 年增长了 24%,已相当于所有互联网流量的 65%.随着互联网更深入地融入社会生活,信息来源变得更加丰富和复杂,一些含有不当内容的有害视频极易被快速传播并渗透进社会生活各个方面,因此,对互联网上传输的海量有害视频进行快速识别是对网络空间有效管理的必要前提.同时,如何在维护这一需求的同时,保护用户隐私并支持信息的自由分享,也成为现代社会面临的关键问题.

目前业界对视频内容进行识别的方法大多通过视频平台对图像进行识别.视频平台在进行内容审核时,往往采用人工审核或人工智能(Artificial Intelligence, AI)识别两种方式.人工审核指的是审核员根据平台的政策和规定通过观看视频的方式进行审核,人工审核方式不仅工作量大,耗时长,并且很难实现快速识别.AI 识别[1-3]是指利用深度学习方法,通过计算机视觉技术实现对视频中的目标对象、场景或行为的精准检测与分类.使用深度学习方法进行视频识别时,为了确保识别的准确性,不仅需要规模足够大的样本数据集,而且需要这些数据包含不同识别内容的多样化标签,同时,深度学习过程需要大量的存储空间和计算资源,这些原因导致中小型视频平台难以承担使用 AI 识别进行视频审核的成本.此外,立足于多视频平台全网协作的视频审核方案很难在实际中部署.

近年也有研究提出了从传输视频的网络流量中识别出有害视频的方法,这类方法不需要多方协作,只要在主干接入点部署流量采集点就可以应用,具有很强的实用性.这类方法需要事先建立公害视频特征库,对网络进行监测时,通过对流量的分析提取被传输的视频特征,与公害视频特征库中的视频特征进行匹配识别.当视频特征库被准确构建后,这类方法的核心技术是如何从传输流量中准确提取出与视频内容相关的特征用以匹配.

视频内容的特征最直接地由应用层数据的特征表达.随着视频平台普遍使用端到端加密,基于流量分析进行视频识别的关键技术难点是视频的应用层内容被加密,并被分为很多 IP(Internet Protocol)报文传输,导致难以直接从密文数据中提取出与视频内容相关的特征.

为了从加密流量中得到与视频内容相关的特征,现有研究利用了 HTTP/1.1 传输视频时 TCP 的头部信息和传输模式.HTTP/1.1 基于 TCP 传输,利用 TCP 头部的响应序列号可将属于同一个视频片段的加密报文的载荷长度相加[4-8],以此作为应用层音视频片段的长度特征.但是这些方法没有考虑到视频传输过程中加密协议和传输协议增加的信息对长度造成的干扰,从而导致无法在大型指纹库场景中准确识别视频.近年一些研究[9,10]提出基于数据加密传输的原理,将从加密流量中提取的视频片段载荷长度更加精准地还原为视频片段本来的长度,可以提高视频识别的准确性,减少误判率.这类方法基于数据加密传输的领域知识提取了精简的特征,只需要较小规模的样本数据集就可以通过机器学习准确地将密文的长度特征还原为视频片段的长度,当领域知识有所改变时,其特征提取方法需要相应地进行调整.

但是,最新的 HTTP/3 协议不再基于 TCP,而是基于 UDP 传输.为了提高数据的传输效率,谷歌公司提出了基于 UDP 的传输层协议 QUIC[11],2022 年 IETF(The Internet Engineering Task Force)基于 QUIC 发布了运行在 QUIC 之上的 HTTP/3[12]协议.根据 W3Techs 的统计显示②,全球前 1000 万个网站中,已有 25.2% 的网站采用 HTTP/3.近年来,国外视频平台如 YouTube、Facebook 和 Instagram 等已广泛采用 HTTP/3 作为其主要的视频传输协议,而国内视频平台仍主要

① 2023 Global Internet Phenomena Report, https://www.sandvine.com/global-internet-phenomena-report-2023.

② Usage statistics of HTTP/3 for websites, https://w3techs.com/technologies/details/ce-http3

使用 HTTP/1.1 和 HTTP/2 协议.然而,HTTP/3 是未来互联网的发展趋势,可以预见未来 HTTP/3 将占有更大的网络流量比例.HTTP/3 使用的 UDP 是无连接的传输协议,头部没有信息可用于提取与视频内容相关的特征.此外,QUIC 协议加密了数据包头和载荷中的传输层元数据,使得传输过程几乎没有可用的明文信息,这导致现有依赖 TCP 头部信息的加密视频识别方法都无法应用于 HTTP/3 协议.

随着 HTTP/3 的发展及其在未来网络中可能的主导角色,为了尽早为国内外视频平台使用 HTTP/3 进行视频传输时的内容筛查工作做准备,研究如何从网络流量中识别使用 HTTP/3 传输的加密视频成为网络空间安全管理中亟需解决的问题.该研究不仅对网络安全的维护具有深远影响,也对促进合法视频内容的传播以及防止有害内容的传播等方面起到关键作用.通过填补现有技术的空白,该研究为全球视频平台构建一个更安全的网络环境提供了重要技术支持,具有极大的价值.

本文通过分析 QUIC 及 HTTP/3 协议的特点,结合视频传输机制,提取使用 HTTP/3 传输的加密视频的数据传输和控制信息特征,考虑各层传输协议对应用层长度的干扰,对从流量中提取出的音视频长度进行修正,最终通过特征在视频库中的匹配来实现视频内容识别,并使用 YouTube、Facebook 和 Instagram 平台的真实视频明文信息构成的指纹库进行验证.

本文的主要贡献如下:

(1)通过提取 HTTP/3 传输流量中的控制信息特征和数据传输特征,提出了从加密 HTTP/3 视频流中提取音视频片段组合并对长度进行修正的方法.基于领域知识的密文长度修正方法可以快速构建指纹修正模型,并且修正误差率接近 0,是大规模指纹库中准确识别视频的基础.

(2)面向视频多片段合并分发导致的海量匹配问题,基于明文指纹库设计了键值数据库.将修正后的密文长度在键值数据库中进行匹配,并使用隐马尔可夫模型和维特比算法构建视频识别模型.

(3)使用真实的 YouTube、Facebook 和 Instagram 视频明文指纹库对本方法进行了验证.结果表明,在小型指纹库和大型指纹库上,本文的方法都可以达到很好的识别效果.本文的研究是在 HTTP/3 流中进行加密视频识别的首次成功尝试.

本文第 2 节对相关研究进行介绍,指出目前研究的现状;第 3 节介绍了本研究的原理和关键技术难点;第 4 节为基于 HTTP/3 传输特性的加密视频识别方法;第 5 节通过真实世界 HTTP/3 流量数据验证了本方法的准确性;第 6 节总结了全文.

2　相关工作

随着互联网的快速发展,网络流量的加密和协议的复杂化使得网络流量分析变得更加困难.根据流量识别目标的不同,加密流量分析工作可以分为两类,分别为流量分类和流量细粒度信息识别.流量分类包括通用应用类型识别、异常流量识别、隐蔽流量识别等.流量细粒度信息识别则是对加密流量中承载的细粒度信息进行识别.例如,用户浏览的网站或者网页、用户使用软件时的行为、用户播放视频或者使用视频会议时的服务质量感受、用户观看的具体视频内容等.

由于流量加密技术的广泛应用,现有研究主要使用深度学习方法,也有基于特征工程的机器学习分类方法进行流量分析.现有文献在流量分析中关注的问题主要是特征选择[13]、小样本[14]、无标签数据[15]、不平衡数据集[16]、数据漂移[17]、对流量特征细微变化敏感[18]等人工智能分类算法面临的问题.

随着国际形势的日趋严峻,各国都加大了对网络空间的管理要求,对视频内容或者标题的识别需求是近年我国对网络空间安全管理的迫切需求.在保护正常用户隐私、不解密网络流量数据的前提下,从加密流量中识别传输的特定视频内容是本文的研究目标,属于加密流量分析中的流量细粒度信息识别.

为了能识别出特定内容的视频,首先需针对待识别视频构建一个视频特征库,其中每个视频需具有可识别的稳定特征,通常称为视频指纹,因此视频特征库也称为视频指纹库.由于视频数据在传输时被加密,无法直接从流量中分析出视频内容,现有方法基于动态流媒体技术的特点,从加密传输的视频流量中提取与视频应用层相关的载荷长度序列特征,并与视频指纹库中具有特定视频内容标签的视频指纹进行对比,如果流量特征与视频指纹库中的视频指纹匹配,就可以根据视频指纹库中该视频的标签记录获知该视频的内容,从而达到识别视频内容的目的.

根据视频指纹库构建的方法,面向加密流量的视频识别方法可分为基于密文指纹库和基于明文指

纹库两类方法.

基于密文指纹库的方法是指使用视频播放时的名称及同时采集到的加密数据传输特征构建密文指纹库,将从密文流量中提取的传输特征与密文指纹进行匹配.现有文献选取了视频流的突发模式[5]、数据链路层和网络层加密的 Wi-Fi 流量[19-20]、基于差分比特率的特征[6,21]等视频传输特征,使用深度学习模型,在较小规模的数据集中进行分类识别.例如 Bae 等人[22]在 LTE(Long Term Evolution)网络中,将每 0.2 秒的聚合流量长度序列作为卷积神经网络(Convolutional Neural Network, CNN)模型的输入,基于 100 个 YouTube 视频组成的密文指纹库,实现了精度在 0.928 和 0.985 之间的视频识别.

基于密文指纹库的方法都只使用了规模较小的视频库.因为使用密文构建的指纹并不是稳定的特征,视频流的密文传输特征不仅取决于视频内容,也与网络信道的瞬间服务质量相关,这导致用于构建密文指纹库的密文特征本身就具有不稳定性,当视频指纹库增大时,这些不稳定特征造成的误差会影响识别的准确性,因此现有基于密文指纹库的研究进行方法验证时使用的视频数量都在几十到几百之间[5,6,19-22].这使得基于密文指纹库的方法无法适用于包含大量待识别视频的真实场景.

基于明文指纹库的方法是指通过带外方式获取视频片段的明文信息构建明文指纹库,将从密文流量中提取的传输特征进行修正后与明文指纹进行匹配.Reed 等人[4,9]在研究中提出了一种视频识别方法,该方法基于流量报文中的 TCP 头部信息获得视频片段应用数据单元,从而根据事先建立的明文指纹库识别视频.由于从加密流量中得到的视频片段长度包括了 HTTP 协议和 TLS(Transport Layer Security)协议增加的头部信息,这些信息也被混入应用层载荷,因此 Reed 对视频片段长度进行了一些修正,并使用 30 个视频片段作为匹配窗口,最终在规模为 330364 个视频的指纹库中得到了 99.5% 的召回率,但是没有给出其他的测试指标.在同样使用明文指纹库的基础上,Wu 等人[10]对 HTTP/1.1 加密传输的视频片段进行精准还原,从而尽可能减少 HTTP 协议和 TLS 协议头部信息对视频应用数据单元长度带来的误差,提高了识别的准确性,在 20 万级别的模拟视频指纹库中达到了 100% 的识别准确率、精确率、召回率和 0% 的误报率.文献[23]提出了一种易于拓展的加密视频攻击方法 RoVIA,使用序列匹配算法将从加密流量中提取出的视频块大小序列放进视频指纹库进行识别,该方法直接禁用 QUIC 协议,只采集 TLS 流量.

基于明文指纹库的方法使得视频识别可以在大型指纹库场景中得以应用.由于使用明文构建的指纹来自视频片段的元信息,因此是稳定特征,不会受到网络信道服务质量的影响.只要能够从加密流量中精准复原出视频片段的长度,就可以在大规模明文指纹库场景中准确识别视频.影响视频识别准确率的关键技术主要有两个,分别为从加密视频流量中正确提取视频片段信息并尽可能准确复原出视频片段应用层的长度特征,以及将视频片段长度序列与明文指纹库中的视频片段序列进行匹配.

表 1 对已有方法中使用的指纹库大小进行了总结.基于上述分析可见,密文指纹库至多使用数百条视频,相比之下,明文指纹库的规模可达到数十万级,因此只有使用明文指纹库才能实现大型视频指纹库场景中的视频识别.在已有确定的明文指纹库的前提下,如何从传输的加密视频数据中准确提取并复原出应用层长度特征是影响识别准确性的关键.已有研究利用了 TCP 头部信息[9,10]和 TLS[10]协议头部信息,基于 HTTP/1.1 的传输模式将属于同一个视频片段的报文组合进行后续分析.但是这类方法在面对基于 UDP 传输的 HTTP/3 协议时将不再适用,因为 UDP 头部没有类似 TCP 报文的头部信息帮助提取属于同一个视频片段的报文,且 HTTP/3 协议基于的 QUIC 协议将 TLS1.3 进行了封装,也无法提取 TLS 头部信息.此外,HTTP/3 使用的多路复用技术导致多个视频片段混合传输,提取单个视频片段更加困难.

表 1 已有方法中使用的指纹库大小

指纹库类型	密文指纹库			明文指纹库	
指纹库方法	文献[5]	文献[6]	文献[22]	文献[4]	文献[10]
指纹库大小(个)	18	200	300	330364	200000

由于 HTTP/3 是一种 2022 年才被标准化的协议,针对 HTTP/3 视频流量的研究较少,只有少量对早期 QUIC 协议进行的研究.针对 QUIC 的研究主要有两类:一类是直接利用传输层信息进行粗粒度分类,另一类是基于传输层信息进一步还原应用层信息,进行更细粒度的应用层信息识别.

粗粒度分类是指直接利用传输层信息对是否为视频流进行二分类或者是对多个视频类别进行多分类.文献[24]提出了一种方法 Silhouette 用于识别 YouTube 视频流.该方法使用传输层流统计信息作

为特征,能识别给定流是否为 YouTube 视频流. Silhouette 算法可用于 HTTPS(Hypertext Transfer Protocol Secure)和 QUIC 的视频流. 二分类的结果粒度过粗,也有研究对视频所属的类别进行分类. 文献[25]根据视频内容的性质将规模为 3036 个视频的数据集分为十二个类别,从视频流中提取包括基于报头和有效载荷的特征进行分类,此方法可应用于将 QUIC 协议传输的视频进行类别分类. 文献[26]对视频播放的体验质量(Quality of Experience, QoE)指标进行三个类别的分类,作者利用了传输层特征和到达间隔时间、数据包大小等网络层特征来训练机器学习分类器,该方法在 HTTPS 协议下达 90% 的分类准确率,QUIC 协议下达 85%的分类准确率.

上述粗粒度识别只能对视频进行分类,无法识别视频的细粒度特征. 细粒度的视频识别需要从加密流量中精准还原出应用层的特征. Xu 等人[27]根据加密流量中的视频块下载信息推断移动视频自适应行为,该方法首先需要从与视频块相关联的一组加密数据包中估算块大小并基于视频库推断对应视频块的信息,最大误差在 HTTPS 下为 1%,在 QUIC 下为 5%. Wu 等人[28]根据 QUIC 视频请求和响应的时间特性推断应用数据单元类型,但是该方法所依赖的视频传输方式已经改变,因此使用该方法已经无法从现有的 HTTP/3 视频流中还原出应用层特征.

对视频内容进行识别属于细粒度识别,目标是基于应用层特征在大规模视频指纹库中精准确定视频内容标签. 由于现有基于 TCP 的加密视频识别技术无法从 HTTP/3 传输的视频流中提取传输视频片段的报文,也无法将使用多路复用传输技术的视频片段组合在视频指纹库进行匹配,目前尚未有对 HTTP/3 视频识别的有效方法.

针对上述问题,本文提出了一种基于明文指纹库,从 HTTP/3 视频流中提取多个音视频片段组合传输数据,并将音视频片段组合的密文长度修正为应用层音视频片段组合长度的方法,其中,“音视频片段组合”是指音频片段和视频片段明文的组合,“音视频片段组合的密文长度”是指当音视频片段组合在网络中加密传输时,各层协议对音视频片段组合进行加密、封装并且切分成多个 UDP 数据包之后的 UDP 包载荷长度之和. 然后,将修正后的长度序列作为 HTTP/3 视频的特征进行视频识别,从而在大型指纹库的场景中实现面向 HTTP/3 视频流的加密视频识别.

3　加密视频识别原理及关键技术问题

3.1　加密视频识别原理

目前从加密流量中识别视频的方法都利用了流媒体的 HTTP 自适应流(HTTP Adaptive Streaming, HAS)技术提取视频流特征. HAS 技术将视频内容切分成多个片段后使用可变比特率(Variable Bit Rate, VBR)[29]技术编码,VBR 可以根据文件内容的复杂程度决定使用不同的比特率编码,同一个视频会被编码为多种比特率和分辨率的组合,因此每个视频片段的长度与视频内容的复杂性相关,也和播放的分辨率相关. YouTube、Facebook 等主流互联网视频平台都采用了 HAS 中使用最广泛的基于 HTTP 的动态自适应流媒体技术[30](Dynamic Adaptive Streaming over HTTP, DASH)提供视频服务. 如图 1 所示,DASH 技术将视频按照播放时间顺序切分成多个片段流式传输,每个视频片段的长度序列与视频内容有关,视频片段的长度序列构成了视频的特征,可以据此实现对特定视频的内容识别. 基于 DASH 技术的特点,通过流量分析对视频进行识别的方法已有一些研究成果[4-7,31].

根据本文第 2 节的现状调研和分析总结,为了在大型指纹库中实现准确的加密视频识别,需要使用明文指纹库的方法,将从信道提取的视频传输特征与明文指纹库进行匹配. 因此本文的视频识别原理如图 1 所示,首先,通过带外方法获取视频明文信息,提取明文信息中的音视频片段长度序列并打上内容标签以构建视频明文指纹库. 其次,从加密数据中提取用户播放视频时的传输数据,提取视频传输特征并进行复原. 最后,将加密音视频片段复原后的应用层特征与明文指纹库进行匹配,即可识别得到视频内容.

本文的应用场景为网络空间中有害视频的识别. 从保护正常用户上网隐私权的角度来看,本文方法只监控特定有害视频,不会侵犯用户观看正常视频的权利.

视频识别的基本原理虽然已经确定,但是现有的方法并不能识别使用 HTTP/3 协议进行传输的加密视频,关键问题在于 HTTP/3 协议为流量分析带来了新的挑战.

3.2　HTTP/3 传输给视频识别技术带来的技术难点

HTTP/3 使用 QUIC 协议为 HTTP 语义提供传输,QUIC 提供协议协商、基于流的多路复用和流

控制.已有基于 TCP 以及 HTTP/1.1 的视频识别方法不再适用 HTTP/3,其原因主要有两点,分别是数据传输模式以及数据单元传输控制信息的差异.

首先是 HTTP/3 的传输模式带来的技术挑战.已有的视频识别方法都利用了 HTTP/1.1 的传输模式,HTTP/1.1 的传输模式中,服务器严格按照请求顺序发送响应数据,因此两个请求之间的数据只能是一个音频或视频片段的响应数据,据此可以根据 TCP 头部的响应序号,从加密数据中将音频片段和视频片段数据分别分离出来,这是现有加密视频识别方法提取音频或者视频应用数据单元的基础.在 HTTP/3 协议中,客户端连续请求音频片段和视频片段时,服务器可以组合传输多个音频片段和视频片段.图 2 展示了视频播放过程中音视频片段组合传输的例子,第 6、7、8 个和第 9、10、11 个视频片段被分别组合传输,第 4、5、6 个音频片段被组合传输.由于响应报文混杂在一起,导致难以将组合传输的音频或视频片段数据单独分割出来,因此本文研究从加密数据中分割出音视频片段组合的方法.

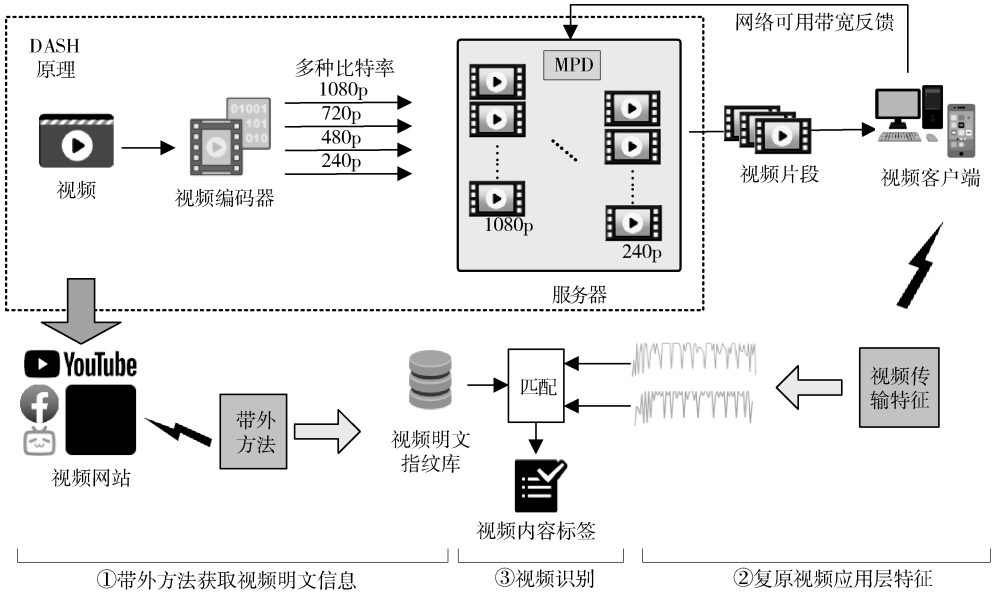

图 1 DASH 技术与视频识别原理

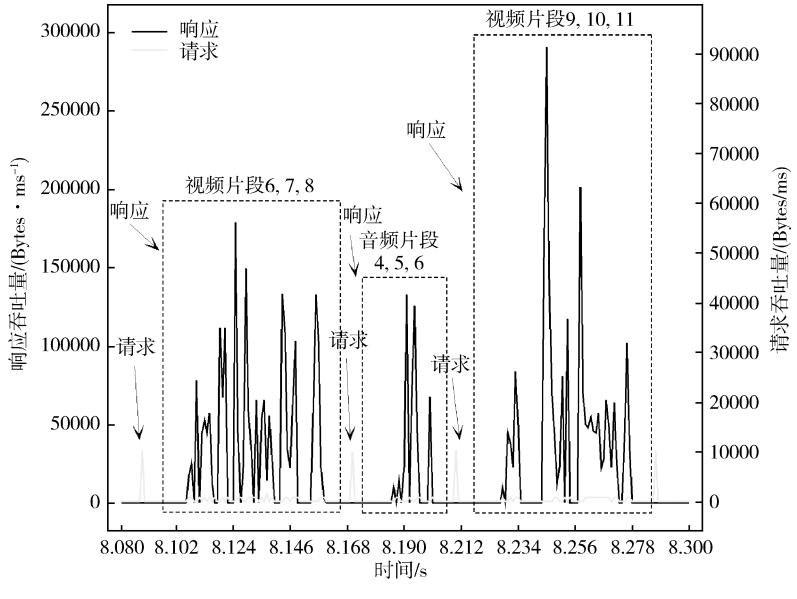

图 2 视频播放过程中视频片段的组合传输

其次,基于 UDP 的 HTTP/3 协议为了保证传输的可靠性,在 QUIC 数据单元中增加了传输控制信息,这些传输控制信息和应用数据混杂在一起加密传输,进一步影响了从密文中提取的视频片段长度特征的准确性. 如图 3 所示,应用层数据单元首先由 HTTP 协议封装,再被切片、封装、加密为 QUIC 数据包. 在每一个 QUIC 数据包中,QUIC 包头与 HTTP 消息间包含三层结构. 第一层是 QUIC 数据包,其头部信息中包含的数据包序号严格递增,解决了窗口阻塞问题. 第二层是 QUIC 帧,其中传输视频数据的主要帧类型为 Stream 帧,通过 Stream ID 进行唯一确认,实现了有序字节流. QUIC 使用 Stream 帧进行端到端的通信,一个或多个 Stream 帧被组装成一个 QUIC 数据包. 当应用数据单元非常大的时候,需要通过多个 QUIC 数据包传输,这些传输同一个应用数据单元的 QUIC 数据包含有同样的 Stream ID. 多个并发传输的应用数据单元,就可以通过不同的 Stream ID 加以区别. 第三层是 HTTP/3 帧,包括 HTTP/3 的头部和应用层载荷,主要包括 HEADER 帧和 DATA 帧,分别传输 HTTP 报头和正文. 这些信息在 QUIC 数据包单元中加密后无法进行区分,本文需要对 QUIC 数据传输的这些控制信息进行分析,通过修正算法减少控制信息给数据特征提取带来的干扰.

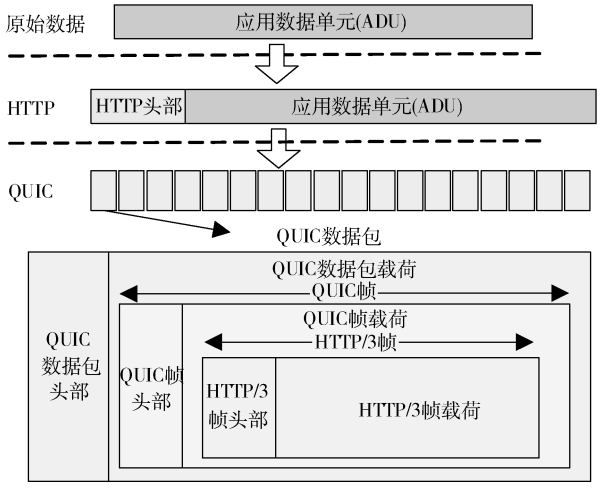

图 3　QUIC 数据包中包含的控制信息和应用载荷信息

因此,HTTP/3 的传输机制给视频识别带来了两个技术挑战:第一是如何分割出多个音视频片段组合的问题,第二是如何解决 HTTP/3 基于的 QUIC 数据传输机制增加的控制信息对还原音视频片段组合长度带来的干扰.

4　研究方法

4.1　方法概述

本文提出了一种从加密 HTTP/3 流中识别视频内容的方法. 方法框架如图 4 所示.

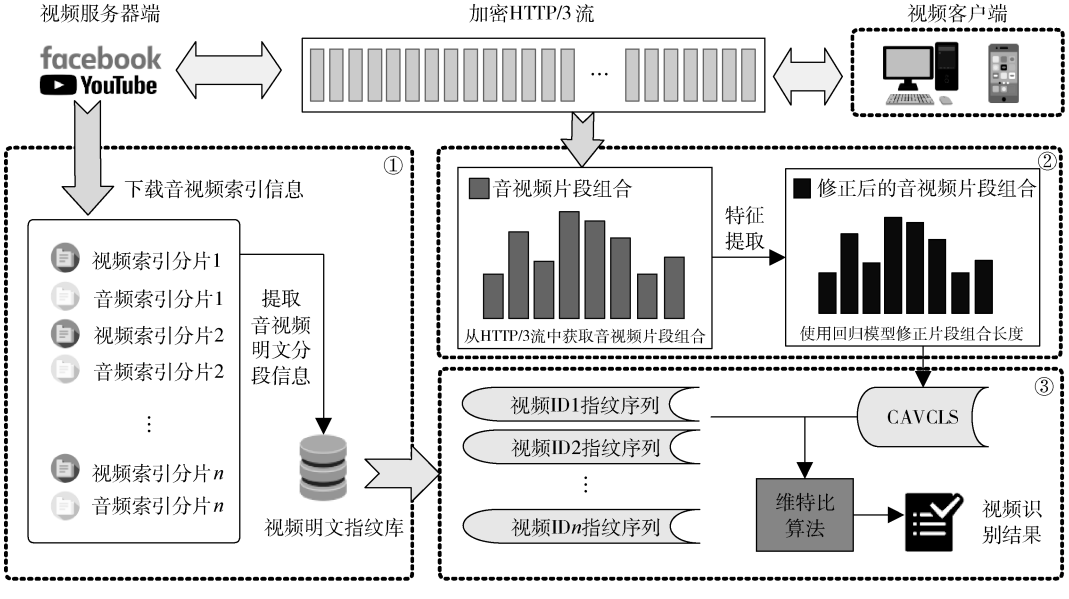

图 4　方法总体框架

首先,建立视频明文指纹库. 使用带外方式从视频平台服务器中得到音视频片段索引信息,根据索引信息下载对应的音频和视频索引片段,从而提取视频的明文指纹信息并给出对应的内容标签.

其次,对需要识别的 HTTP/3 视频流量,分析视频数据并提取音视频片段组合的密文长度特征.

由于传输数据中增加了协议头部信息等字段,此时获取的密文长度相较原始视频明文指纹组合要长许多,因此,还需要对密文长度进行修正.

为了使还原后的密文长度逼近原始音视频片段组合长度,本文使用训练数据中提取的音视频片段组合对应的明文指纹组合长度标记训练数据后,结合提取的特征使用机器学习来修正密文长度.

最后,使用修正后的密文长度序列与明文指纹库进行匹配,得到最终识别的视频内容.

4.2 从 HTTP/3 流中提取音视频片段组合长度特征

加密视频的识别是通过还原后得到的应用层音视频片段组合长度特征与指纹库中的视频明文长度进行匹配,因此首先需要从 HTTP/3 流中提取音视频片段组合的密文长度特征. HTTP/3 采用了多路复用的传输机制,但是多路复用技术的复用和解复用过程也会占用服务器和客户端的资源,所以并非所有的数据都会使用多路复用混杂传输. 在视频播放的开始阶段,为了使得客户端能尽快播放画面和声音,视频片段和音频片段会使用多路复用技术混合传输,但是在后续播放阶段,为了节约服务器和客户端的资源,视频数据一般是由音频或视频片段组合随机交替传输.

由于国内视频平台暂未开始大规模使用 HTTP/3,本文采用的数据来源为国外主流视频网站 YouTube、Facebook 和 Instagram.

分析 YouTube、Facebook 和 Instagram 的视频流量可知,最新的 DASH 使用了多片段组合传输技术. 这些视频平台往往一次性连续请求多个音频和视频片段,服务器响应时会传输这些片段的组合. 一般情况下,音频片段会比视频片段小很多. 图 5 显示了 DASH 机制下的音视频传输的一个实例,Vi 表示第 i 个视频片段,Aj 表示第 j 个音频片段. 客户端首先请求音频数据和视频数据{$V1$,$V2$,$V3$,$V4$,$V5$,$A1$,$A2$}来确保视频可以正常播放,在一段时间后,客户端不再需要同时获取音视频片段,而是根据需要交替请求音视频数据. 随着网络状况的波动,客户端会根据网络情况请求不同分辨率的视频片段.

在图 5 的视频数据传输过程中,一个请求会得到对应的音视频片段组合响应数据,这些响应数据是由特定的音视频片段组合而成,例如{$V9$,$V10$,$V11$,$V12$}、{$A6$,$A7$、$A8$}等,这些音视频片段组合的传输数据大小就是需要提取的音视频片段组合密文长度特征. 本文使用以下三个步骤进行提取.

(1)在视频客户端与服务器交互的过程中获取加密 HTTP/3 视频流量

根据五元组(源 IP 地址,源端口,目的 IP 地址,目的端口,传输层协议)提取 HTTP/3 双向流,并设置阈值筛选出 HTTP/3 加密视频流量.

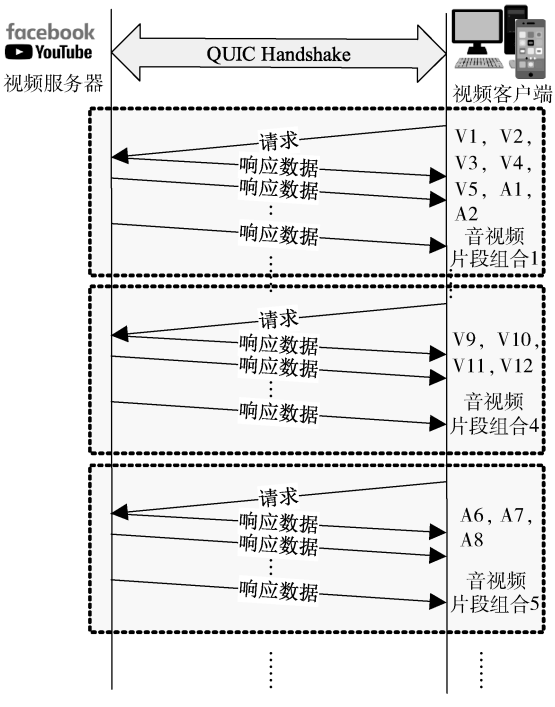

图 5 HTTP/3 视频传输过程

(2)从 HTTP/3 视频流中分割出音视频片段组合序列

由图 5 可见,客户端请求之间的数据为多个音视频片段的组合,只要识别出请求报文就可以将音视频片段组合作为一个单元分割出. 但是在传输中,客户端除了请求,也会发送确认数据,而且数据内容都被加密,难以区分请求和确认报文. 因为请求报文包含了请求的应用层音视频片段组合的描述信息,数据内容远多于确认报文,本文使用数据包的长度区分客户端发出的请求和确认报文. 根据对 HTTP/3 流的分析,因为要携带请求的目标参数,请求数据包的长度一般为 1000 字节左右,而确认数据包只有几十字节. 根据以上特征,一个音视频片段组合的开始标志可以被认定为客户端向服务器发送的长度为 1000 字节左右的数据包. 持续接收一段时间后,如果客户端又发送了一个长度为 1000 字节左右的数据包,这个数据包之前的最后一个响应数据包是这个音视频片段组合的结束. 由此可以将 HTTP/3 流中所有组合传输的音视频片段组合分割出.

(3)从分割出的音视频片段组合序列中提取密文长度特征序列

使用步骤(1)、(2)遍历一个 HTTP/3 视频流,

可以分割出多个音视频片段组合单元.对于分割出的每个音视频片段组合单元,将所有响应数据包的 UDP 载荷长度相加得到这个音视频片段组合的密文长度,由此就可以得到该 HTTP/3 视频流的音视频片段组合密文长度特征序列.

需要注意的是,因为音频片段和视频片段会组合在一起加密传输,并且这些组合传输的片段是无法分割出的,因此本文提取的是音视频片段组合而非音视频单个片段的特征.

4.3　密文长度特征修正

由于在加密传输过程中加入了传输和加密的控制信息,音视频片段组合传输的密文长度特征与对应的明文指纹组合长度存在偏差,如果不进行长度修正,直接用密文长度序列进行匹配会导致很大的匹配误差.因此从 HTTP/3 视频流中提取的密文长度特征必须进行修正才能还原得到应用层特征,用于视频识别.

4.3.1　明文指纹长度与密文长度特征的差异比较

本方法面向真实视频平台,在分析过程中使用目前全球最大的视频平台 YouTube 的数据,另外,本文采集了 Facebook 和 Instagram 的视频数据用于验证方法的通用性.

本文使用 YouTube 的数据对明文指纹组合长度和密文长度特征的差异进行比较.图 6 为 YouTube 平台部分音视频片段组合长度分布图,如图 6 所示,视频片段组合长度大多集中在 0.834MB 至 2.522MB 之间,中位数为 1.695MB,音频集中在 192.079KB 至 649.910KB,中位数为 426.604KB.但是这些原始片段组合在传输的时候必然加上了控制信息,从密文中提取的音视频片段组合的密文长度会超过原始片段组合长度.

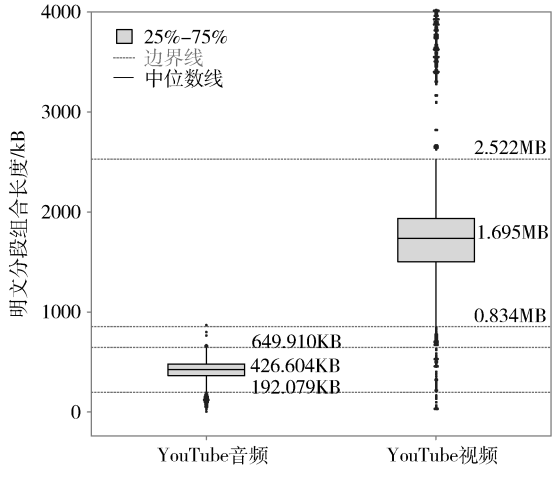

图 6　YouTube 原始音视频片段组合长度分布

在提取 YouTube 密文长度序列之后,将其与视频明文指纹库中对应的原始片段组合序列进行对比.原始音视频片段组合与对应密文的差值,即原始残差如图 7 所示.对于 YouTube 平台的视频,视频片段组合部分的原始残差的长度分布集中于 19.007KB 至 45.470KB,中位数为 36.521KB;音频残差集中在 4.153KB 至 14.196KB,中位数为 9.298KB.使用 Y_v_Resi 表示视频原始残差中位数,Y_v_Plain 表示相应明文长度中位数,Y_a_Resi 表示音频原始残差中位数,Y_a_Plain 表示相应明文长度中位数.视频片段误差率 Y_v_Resi/Y_v_Plain 约为 2.10%,音频片段误差率 Y_a_Resi/Y_a_Plain 为 2.18%,音频和视频片段的误差率较为相似.

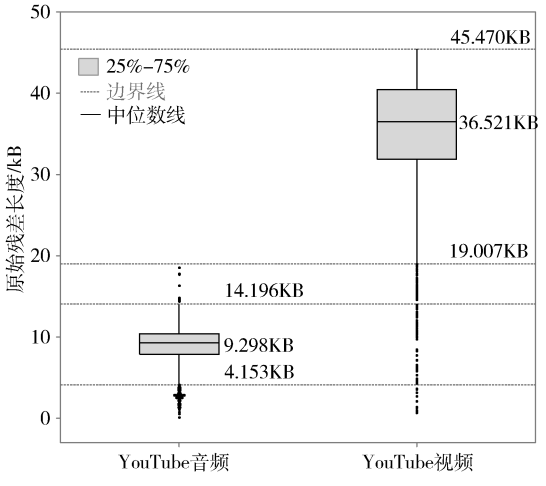

图 7　YouTube 音视频片段组合原始残差长度分布

如果不进行修正,从音视频片段组合中提取的密文长度平均会比原始音视频片段组合长度长 2.1%左右.当传输分辨率较高的视频片段时,绝对偏差会很大,导致无法在指纹库中准确匹配原始片段组合.因此,需要对从音视频片段组合中提取的密文长度进行修正.

4.3.2　基于 HTTP/3 控制信息对密文长度特征的修正

本文首先根据领域知识提取视频传输特征,再使用机器学习对密文长度进行修正.

为了能够对密文长度特征进行修正,必须尽可能将加密传输过程中添加的辅助控制信息长度从密文长度中减去,因此必须对加密传输过程进行分析,确定增加的控制信息长度.

在传输过程中,原始音视频片段组合首先基于 HTTP/3 协议规范切片,每一片加上 HTTP/3 帧

头部信息后再次被切片、封装为 Stream 帧，最后被封装、加密为 QUIC 数据包，其中包含的控制信息就是在长度修正过程中需要减去的长度信息. 因为这些控制信息是为了加密和传输添加的，每次加密和传输的时候都可能随着网络环境，参数配置等发生变化，只有将这些可变长度从密文长度中减去，才能尽可能准确还原出视频片段组合的明文长度，用于视频的识别.

本文通过分析加密及传输过程增加的控制信息确定需要减去的控制信息长度. 如 3.2 节给出的图 3 所示，每个 QUIC 数据包、Stream 帧及 HTTP/3 帧的结构相同，需要去除的头部信息长度也大致相同. 除了一些固定字节的信息之外，在 QUIC 数据包中，变长信息包括在 QUIC 数据包头部的 Packet Number；在 Stream 帧头部中的 Stream ID、Offset；在 HTTP/3 帧头部中的 Length. 下面分别分析 QUIC 数据包头部的 Packet Number，Stream 帧头部中的 Stream ID、Offset，以及在 HTTP/3 帧头部中的 Length 字段占用的字节数：

(1)Packet Number 被 QUIC 用于实现可靠传输. 由于 Packet Number 严格递增，只有在一个 QUIC 流中的前面少量数据包中占用 1 字节，大部分都是长整型，因此 Packet Number 在绝大部分 QUIC 数据包中占用 2 字节；

(2)Stream ID 起到区分 Stream 的作用，最低两位用于标识流发起者和流方向. 由于用于传输视频数据的 Stream 的最低两位取值都为 0，因此其 Stream ID 的取值从 4 开始以 4 为倍数递增，前 15 条 Stream 中 Stream ID 为 4 至 60，占用 1 字节，后续 Stream 中的 Stream ID 都大于 64，占用 2 字节，与前者相差 1；

(3) Offset 表示该 Stream 帧中的载荷在 Stream 中的偏移量，根据实际数据的统计，响应数据包大小大多在 1000 字节以上，Offset 的取值跨度较大，仅有少量 Offset 字段占用 1 或 2 字节，其余占用 4 字节；

(4)Length 表示 HTTP/3 帧有效载荷的长度，使用除最高一位的其余位表示长度数据. 在本文中 HTTP/3 帧有效载荷作为音视频响应数据量较大，绝大部分占用 4 字节，少部分占用 1 或 2 字节.

基于上述原理，可以筛选出影响密文长度的重要因素. Packet Number、offset、Length 字段在数据包中占用的字节数较为恒定，在修正时可以不作考虑. 而随着视频长度的增加，传输的音视频片段组合超过 15 个后，Stream ID 字段对密文长度的影响也会增大. 假设 Stream ID 为 60 的 stream(即第 15 条 stream)中包含 9295 个数据包，Stream ID 为 64 的 stream(即第 16 条 stream)中包含 10869 个数据包，前者的 Stream ID 字段在其整个 stream 中共占用 9295 个字节，后者共占用 21738 字节，若不考虑 Stream ID 字段的影响，将其占用的字节都计为 1，则对于后者来说，修正产生的偏差将会达到 10869 字节. 因此，本文将 Stream ID 字段作为修正密文长度的重要控制信息特征.

服务器发送响应数据时，根据一个 Stream 响应开始的不同，有两种传输情况. 第一种情况是先将响应信息与 HTTP/3 HEADER 帧分为两个较小的包传输，然后再传输视频数据的第一个 QUIC 包；第二种情况是将响应信息单独以一个 100 字节左右的 QUIC 包传输，然后将 HTTP/3 HEADER 帧与部分视频数据合并为一个 QUIC 包传输. 由于本文是从包含视频数据的第一个 QUIC 包开始累加载荷长度来获取音视频片段组合的密文长度，因此对第一种情况，可以直接从该 Stream 的第三个 QUIC 包开始减去增加的 QUIC 控制信息，而对第二种情况，第二个 QUIC 包中除了 QUIC 头部信息，还需要再减去 HEADER 帧的长度. 因此，本文将是否需要减去 HEADER 帧的长度作为第二个控制信息特征.

需要注意的是，部分视频平台如 Facebook 为了模糊音视频片段长度，其传输过程中产生的 QUIC 数据包中还可能存在填充帧，由于填充帧长度不定，因此本文不将其作为长度修正的特征.

为了修正密文长度，还需提取数据包数量和载荷长度这两个数据传输特征. 根据以上对 QUIC stream 和数据包中控制信息的分析，本文提取了以下修正时需要考虑到的特征：

(1)$PACKET_{count}$：一条 Stream 中的响应数据包数量.

(2)$STREAM_{len}$：一条 Stream 中的所有数据包 UDP 载荷长度之和.

(3)$STREAM\_ID_{flag}$：Stream ID 取值是否大于等于 64. 大于等于 64 取 1，否则取 0.

(4)$MINUS_{flag}$：是否需要减去 HEADER 帧的长度. 需要减去取 1，否则取 0.

统计密文长度与视频明文指纹库中对应的明文指纹组合的比值，发现呈线性关系，因此可以使用多元线性回归方法修正密文长度. 上述 $PACKET_{count}$、

$STREAM_{len}$、$STREAM\_ID_{flag}$、$MINUS_{flag}$ 四个特征被用于产生多元线性回归模型,其中,数据传输特征 $PACKET_{count}$、$STREAM_{len}$ 作为自变量,对应的明文数据 L_{fit} 作为因变量. 当 $STREAM\_ID_{flag}$ 取不同值时,修正公式如下:

$$L_{fit}=\begin{cases}\alpha\cdot STREAM_{len}-\beta\cdot PACKET_{count}-\gamma, & STREAM\_ID_{flag}=0\\ \alpha\cdot(STREAM_{len}-PACKET_{count})-\beta\cdot PACKET_{count}-\gamma, & STREAM\_ID_{flag}=1\end{cases} \tag{1}$$

公式(1)中,α 表示一个 Stream 的所有数据包中,Stream ID 这个控制字段所占用的长度对密文长度的影响,根据对协议的分析,Stream ID 取值大于等于 64 时,每个数据包中的 Stream ID 会多占用 1 个字节,因此需要额外减去数据包数量的字节数再做拟合. β 表示 QUIC 数据包头部的 Packet Number、Stream 帧头部的 offset 等存在于每个数据包中的字段对密文长度的影响,由于这些字段长度是相对不变的,因此可以合并使用一个参数. γ 表示 HTTP/3 帧头部中的 Length 等较少出现的字段对密文长度的影响. 这三个参数的值通过数据集拟合获得.

接着,对 $MINUS_{flag}$ 取 1 和 0 时根据公式(1)分别进行拟合. 表 2 展示了使用采集的 YouTube 视频流量进行训练后得到的系数值和评估指标,其中 *MSE* 为均方误差,*RMSE* 为均方根误差,*MAE* 为平均绝对误差,这三个指标可以评价数据的变化程度,值越小说明回归模型具有更高的精确度,*R-squared* 为拟合度,越接近 1 表示模型拟合度越高. 从表中可以看出,当 $MINUS_{flag}=1$ 时,对指纹的修正产生的误差较大.

表 2　训练后模型系数及评估指标

类型		YouTube 视频	
		$MINUS_{flag}=0$	$MINUS_{flag}=1$
系数	α	1.000007	1.000462
	β	28.016395	28.627087
	γ	−22.253024	37.008522
评估指标	*MSE*	21.93	1530.46
	RMSE	4.68	39.12
	MAE	3.31	35.70
	R-squared	1	1

使用训练后的模型对密文长度序列进行修正后,残差分布如图 8 所示.

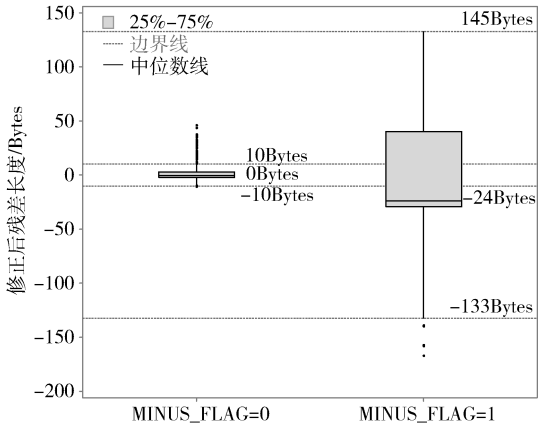

图 8　YouTube 音视频片段组合修正后残差长度分布

根据 $MINUS_{flag}$ 的取值,YouTube 平台的密文指纹修正后残差的分布分为两种情况. 利用四分位数间距(Inter Quartile Range, IQR)规则可以计算数据集整体的离群值边界,IQR 为上四分位与下四分位的差值,利用 IQR 的 1.5 倍为标准,将残差的范围选定在[下四分位 − 1.5IQR, 上四分位 +1.5IRQ]区间内. 当 $MINUS_{flag}=0$ 时,该区间为[−10, 10]字节,97.82% 的数据被认定为修正正确,当 $MINUS_{flag}=1$ 时,该区间为[−133, 145]字节,99.36% 的数据被认定为修正正确. 不在区间范围内的,即被认定为未正确修正的数据,主要由采集环境不够稳定导致丢包而产生的. 对指纹进行修正后,$MINUS_{flag}=0$ 时,视频误差率为 0.0000046%,音频误差率为 0.0001118%. $MINUS_{flag}=1$ 时,视频误差率为 0.0000171%,音频误差率为 0.0054679%,相比原始残差大大降低且几近于 0.

由统计结果可见,经过修正后得到的音视频片段组合的密文长度非常接近原始明文指纹组合长度. 修正后的密文长度序列将用于视频识别,在后文中,本文将修正后的单个音视频片段组合长度称为 CAVCL(Corrected Audio/Video Combination Length),修正后的音视频片段组合长度序列称为 CAVCLS(Corrected Audio/Video Combination Length Serials).

需要说明的是,上述系数、残差的值等由本文采集的 YouTube 平台数据样本确定,但是方法具有通用性. 本文实验部分将使用 Facebook 和 Instagram 平台数据进行方法通用性验证.

4.4　指纹匹配和视频识别

在对密文长度特征进行修正之后,将得到的 CAVCLS 与视频明文数据库中的指纹进行对比得到内容标签,对加密视频的识别分为指纹匹配和视

频识别两个步骤.指纹匹配指将每个 CAVCL 在明文指纹库中进行匹配,由于 CAVCL 是从密文中还原得到的,存在一定的还原误差,所以在大型指纹库场景中,一个 CAVCL 有可能匹配出多个明文指纹组合,相应地导致 CAVCLS 可能匹配出多个待选的音视频明文指纹组合序列.视频识别指从 CAVCLS 匹配结果组成的待选明文指纹组合序列中找到可能性最大的视频,并输出该视频的内容标签.

4.4.1 指纹匹配

指纹匹配首先将 4.3 节修正出的单个 CAVCL 按时间顺序组合成 CAVCLS. CAVCLS 主要与 DASH 视频分发机制有关,也会受到视频平台实现机制的影响.本文将 YouTube 这个较早且大规模使用 QUIC 和 HTTP/3 协议的平台作为实例,对它的视频传输机制进行分析.

(1)数据分发机制:YouTube 在视频播放开始时,因为画面需要同步声音,为了能尽快播放,使用了多路复用技术,存在音视频片段同时传输的现象.在后续视频播放时,客户端连续请求多个视频或者音频片段,这些片段会被组合传输,这时因为已经接收的音视频数据正在播放,为了减少多路复用在服务器和客户端的资源耗费,非必要时,音频和视频片段往往不会混合传输,而是交替分别传输.由于视频服务商并没有公开视频服务的实现细节,因此本文统计了采集到的播放数据,其多路复用情况如图 9 所示,其中音频片段所占比例集中在 15%左右,视频片段所占比例集中在 10%左右,后续的大部分音视频片段并不会混合传输.

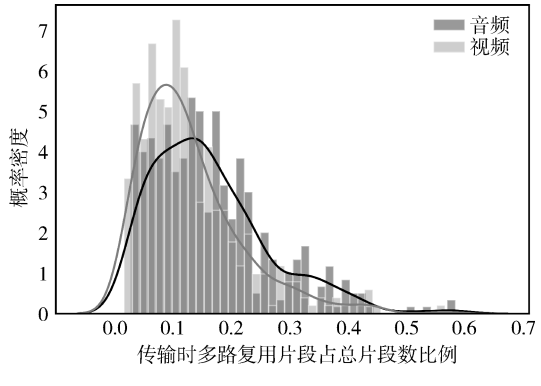

图 9 多路复用片段占视频整体传输比例统计

(2)音视频片段传输特征:如图 10 所示,音视频片段在传输时,因为请求音视频片段组合长度的上限阈值为 2MB,当长度在阈值之上会出现切分后传输的现象,即实际传输过程中,大于 2MB 的音视频片段组合会分片传输,且最后一个片段组合会小于 2MB.因此,若 CAVCL 为 2MB,在匹配时要加上后续的一个 CAVCL,以此类推,直到后续的 CAVCL 小于 2MB 为止.对一个视频传输实例,会有 n 个 CAVCL 构成 CAVCLS.

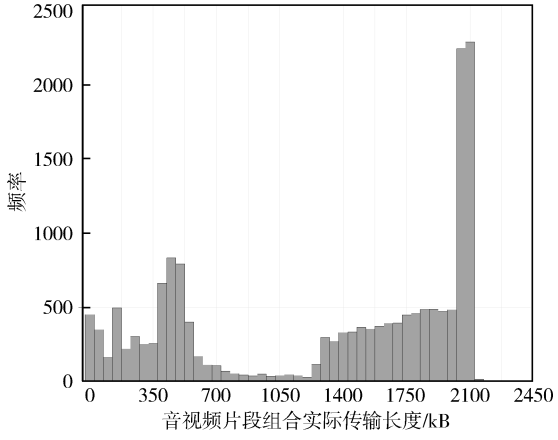

图 10 音视频片段组合实际传输长度

根据上述的视频传输机制,对每一个 CAVCL 进行匹配,得到可能产生 CAVCLS 的明文指纹组合序列.匹配方法如图 11 所示,以一个待匹配视频 CAVCLS 中的第一个 CAVCL 为例,首先,根据 4.3 中图 8 所显示的音视频片段组合修正后残差长度分布设定匹配区间为$[m_1, m_2]$,即 $MINUS_{flag}=0$ 时,在明文指纹库中匹配的区间最小为$[CAVCL_1-10, CAVCL_1+10]$,$MINUS_{flag}=1$ 时,该区间最小为$[CAVCL_1-133, CAVCL_1+145]$,音视频片段组合将基于该区间在指纹库中进行匹配,为了减小误差,可以设置多倍残差范围进行匹配.其次,由于服务器端一次响应多个音视频片段的数据,CAVCL 中可能包含一个视频中多个音视频片段组合的长度,因此需要对于明文指纹库中每一个视频实例,分别使用其中的 1 至 8 个相邻指纹生成组合指纹库.最后,从组合指纹库中取出组合指纹,通过将 $CAVCL_1$ 与组合指纹进行匹配筛选出匹配区间范围内的明文指纹组合长度,并记录对应的视频 ID(Identity Document)、分辨率信息、匹配区间以及包含的明文序列标号.在组合指纹库中,存在一些音视频片段组合长度非常接近的现象.一个 CAVCL 可能匹配到若干个组合指纹,将组合指纹及其记录的信息记为 fc(fingerprint combination),如图 11 中,组合指纹 1 和组合指纹 3 是 $CAVCL_1$ 匹配到的第一和第二个 fc,分别记为 $fc_{1,1}$ 和 $fc_{1,2}$.将 CAVCL 匹配到的所有组合指纹记为 MR(Matching Results),第 n 个 CAVCL 匹配到的所有组合指纹结果记为 MR_n:

$<fc_{n,1}, fc_{n,2}, \cdots\cdots, fc_{n,x}>$,$x$ 为 $CAVCL_n$ 匹配到的所有 fc 数量.

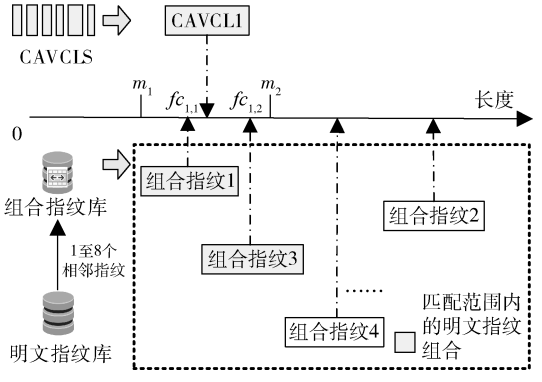

图 11　指纹匹配方法

对整个 CAVCLS 进行上述操作后,得到匹配结果{MR_1:$<fc_{1,1}, fc_{1,2}, \cdots\cdots, fc_{1,k}>$,$MR_2$:$<fc_{2,1}, fc_{2,2}, \cdots\cdots, fc_{2,l}>$, ……, MR_n:$<fc_{n,1}, fc_{n,2}, \cdots\cdots, fc_{n,x}>$},其中 k、l、x 分别为匹配到的不同 fc 数量,n 为 CAVCLS 的长度.

对于指纹匹配中使用的组合指纹库,由于视频平台将多个音视频片段合并传输,成倍增大了待匹配明文指纹组合的规模,大大增加了整体的计算量和匹配时间,因此为了加快匹配速度,本文使用了键值数据库,将数据存储在内存中用于快速读写.键值数据库是一种基于键值对存储数据的数据库系统.它以键和值的形式存储数据,其中键是唯一的标识符,值则是与该键相关联的数据.如图 12 所示,首先,计算出所有可能的组合长度,并建立一个键值数据库,其中每个键对应一个组合长度.然后,对于每个键,将长度等于该键的所有组合对应的信息(例如视频 ID、分辨率、组合片段等)存储在该键的值中.这样,当需要匹配某个组合时,只需要从数据库中查找对应长度的键值,即可快速获取该组合的信息.除此之外,本文使用多数据库并行方法进一步减少匹配时间.

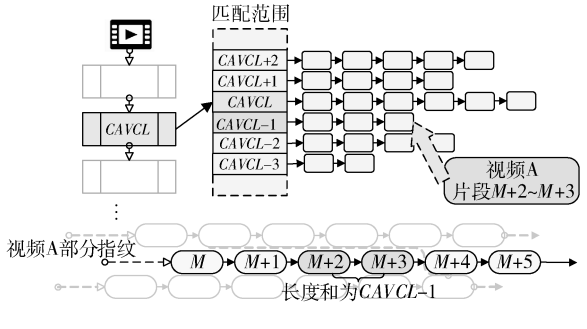

图 12　键值数据库原理

4.4.2　视频识别

在进行初步的匹配后,一个 CAVCL 可能会得到多个匹配结果,因此在进行视频识别时,需要筛选出每个 CAVCL 对应的最有可能的 fc.本文需要将 CAVCLS 与明文指纹库中所有的明文指纹组合序列进行匹配计算得到可能性最大的视频内容标题,实际上这是一种序列相似度的计算.目前已有的方法主要包括编辑距离算法、最长公共子序列算法、动态时间规整算法以及隐马尔可夫模型.

本文在选择视频识别模型时考虑了多种因素,例如时间和空间复杂度以及实时分析的需求等.与其它序列对比算法相比,隐马尔可夫模型(Hidden Markov Model, HMM)的解码问题与本文应用场景类似,其通过建立隐含状态转移概率矩阵和观测状态转移概率矩阵,可以处理序列中的间隔和插入缺失等问题,由于将序列的对比问题转化为了概率计算问题,因此避免了动态规划中计算量的增长问题,适合于比对长度较长的序列,并且可以通过并行计算和分布式计算进行加速.因此,本文使用 HMM 描述视频识别模型,并使用求解解码问题的常用算法维特比算法[32]计算得到一个最有可能的 fc 序列.

本文中,HMM 作为一种统计模型,用于描述视频传输时实际视频明文指纹组合序列随机产生 CAVCLS 的过程.HMM 可以使用 5 个元素进行描述,包括 2 个状态集合和 3 个概率矩阵.其概念及在本文的应用场景中的含义如下:

(1)隐含状态 S:无法被直接观测的隐含状态.在本文中指视频 fc,在经过初步的匹配后,可以使用 CAVCLS 的明文匹配结果 MR{MR_1:$<fc_{1,1}, fc_{1,2}, \cdots\cdots, fc_{1,k}>$, MR_2:$<fc_{2,1}, fc_{2,2}, \cdots\cdots, fc_{2,l}>$, ……, MR_n:$<fc_{n,1}, fc_{n,2}, \cdots\cdots, fc_{n,x}>$}将 S 范围缩小.

(2)可观测状态 O:与 S 相关联,且可以直接观测得到的状态.在本文中指对密文序列进行修正后的 CAVCLS{$CAVCL_1$, $CAVCL_2$, ……, $CAVCL_n$}.

(3)初始状态概率矩阵 $\boldsymbol{\pi}$:代表 S 在初始时刻的概率矩阵.在本文中指开始进行视频识别时,首个可用于识别的 CAVCL 对应各个匹配结果的概率,即对于 $CAVCL_1$,若{$P(fc_{1,1})=P_1$, $P(fc_{1,2})=P_2$, ……, $P(fc_{1,k})=P_k$},则 $\boldsymbol{\pi}=\{P_1, P_2, \cdots\cdots, P_k\}$.

(4)隐含状态转移概率矩阵 $\boldsymbol{A}$:代表 S 之间的转移概率.在本文中指 fc 之间的转移概率.若 $A_{ij}=P(fc_{y,j} \mid fc_{x,i})$,其中 $1 \leqslant x < y \leqslant n$, $1 \leqslant i \leqslant len(MR_x)$,$1 \leqslant j \leqslant len(MR_y)$,则表示当 fc 为 $fc_{x,i}$

时，接下来是 $fc_{y,j}$ 的概率是 A_{ij}.

(5)观测状态转移概率矩阵 $\boldsymbol{B}$：设 $\boldsymbol{S}$ 数量为 n，O 数量为 m，$\boldsymbol{B}$ 代表在该时刻，隐含状态为 $S_i(1\leqslant i\leqslant n)$时，观测状态为 $O_i(1\leqslant j\leqslant m)$的概率. 在本文中指各 fc 对应当前 CAVCL 的概率. 若 $B_{ij}=P(CAVCL_j \mid fc_{j,i})$，其中 $1\leqslant i\leqslant len(MR_j)$，$1\leqslant j\leqslant n$，则表示当前时刻，当 fc 为 $fc_{j,i}$ 时，对应的 CAVCL 是 $CAVCL_j$ 的概率为 B_{ij}.

将每个可观测状态的隐含状态展开得到如图 13 所示的篱笆网络. 在该 HMM 中，圈表示每个 CAVCL 的匹配结果 fc，每条边表示一个可能的状态转换，开始到 $CAVCL_1$ 匹配结果的边值是初始状态概率，fc 之间的边值是隐含状态转移概率，各个 CAVCL 的匹配结果与该 CAVCL 之间的箭头代表观测状态转移概率. 例如，状态 $fc_{1,1}$ 到 $fc_{2,1}$ 边上的取值为 $P(fc_{2,1} \mid fc_{1,1})$，表示 $CAVCL_1$ 的匹配结果是 $fc_{1,1}$ 时，转移到 $CAVCL_2$ 的匹配结果 $fc_{2,1}$ 的概率为 $P(fc_{2,1} \mid fc_{1,1})$.

已知可观测的 CAVCLS，该 CAVCLS 进行指纹匹配后得到的结果 MR 和模型参数$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{\pi})$，视频内容的识别是通过计算最大的匹配概率进而选出最有可能产生该 CAVCLS 的隐藏 fc 序列实现的. 由于穷举搜索计算效率非常低，本文使用维特比算法来解决上述问题.

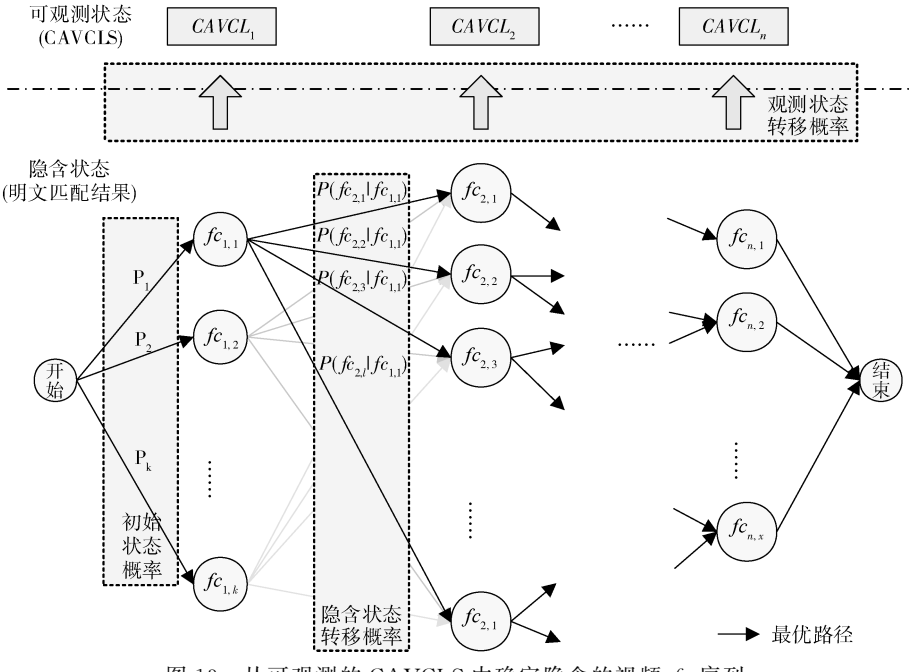

图 13 从可观测的 CAVCLS 中确定隐含的视频 fc 序列

维特比算法被用来解决 HMM 中的解码问题，即用动态规划求解概率最大的路径——最优路径. 维特比算法的核心主要有两点，首先，如果路径是最优的，那么路径上的节点到终点的部分路径也是最优的，其次，假设已知从起点到某个状态的所有节点的最短路径，那么从起点到终点的最短路径必经过其中一条. 根据以上两点，假设从 $CAVCL_i$ 进入 $CAVCL_{i+1}$ 时，从起点到 $CAVCL_i$ 的各个匹配结果的最大概率 fc 序列已经找到，并且已被记录，那么当计算从起点到 $CAVCL_{i+1}$ 的某个匹配结果 $fc_{i+1,k}$ 的最大概率 fc 序列时，只要考虑从起点到 $CAVCL_i$ 所有匹配结果的最大概率 fc 序列，以及 $CAVCL_i$ 所有匹配结果转移到 $fc_{i+1,k}$ 的概率.

在已知可观测的 CAVCLS $\{CAVCL_1, CAVCL_2, \cdots\cdots, CAVCL_n\}$、该 $CAVCLS$ 的匹配结果 $MR\{MR_1: <fc_{1,1}, fc_{1,2}, \cdots\cdots, fc_{1,k}>, MR_2: <fc_{2,1}, fc_{2,2}, \cdots\cdots, fc_{2,l}>, \cdots\cdots, MR_n: <fc_{n,1}, fc_{n,2}, \cdots\cdots, fc_{n,x}>\}$ 和模型参数$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{\pi})$的情况下，本文需要寻找一条 fc 序列$\{f_1, f_2, \cdots\cdots, f_n\}$，使得其在传输过程中产生的 HTTP/3 视频流的 CAVCLS 为$\{CAVCL_1, CAVCL_2, \cdots\cdots, CAVCL_n\}$的概率最大，为了使得公式表达更简洁，以下使用 F 代替 CAVCL.

$\delta_t(f)$ 表示在第 t 个 CAVCL 时，即时刻 t，对应隐含状态为 f 的概率最大值. 计算公式如下：

$$\begin{aligned}\delta_t(f) = \max P(f_t = f, f_{t-1}, \cdots\cdots, f_1,\\ F_t, \cdots\cdots, F_1 \mid (\boldsymbol{A},\boldsymbol{B},\boldsymbol{\pi})),\\ f = fc_{t,1}, fc_{t,2}, \cdots\cdots, fc_{t,N}\end{aligned} \tag{2}$$

以此类推:

$$\delta_{t+1}(f)=\max P(f_{t+1}=f,f_t,\cdots\cdots,f_1,F_{t+1},\cdots\cdots,F_1\mid(A,B,\pi)),$$
$$f=fc_{t+1,1},fc_{t+1,2},\cdots\cdots,fc_{t+1,N} \tag{3}$$

其中 N 为该时刻对应的 MR 的长度.

根据 HMM,$\delta_{t+1}(f)$ 和 $\delta_t(f)$ 之间的递推关系如下:

$$\begin{aligned}\delta_{t+1}(f)&=\max P(f_{t+1}=f,f_t=j,f_{t-1},\cdots\cdots,f_1,F_t,\cdots\cdots,F_1)P(F_{t+1}\mid f_{t+1}=f)\\&=\max(\max_{f_1,f_2,\cdots\cdots,f_{t-1}}P(f_t=j,f_{t-1},\cdots\cdots,f_1,F_t,\cdots\cdots,F_1))P(f_{t+1}=f\mid f_t=j)\\&\quad\cdot P(F_{t+1}\mid f_{t+1}=f)\\&=\max(\delta_t(j)A_{jf}B_{fF_{t+1}}),\\j&=fc_{t,1},fc_{t,2},\cdots\cdots,fc_{t,N},\\f&=fc_{t+1,1},fc_{t+1,2},\cdots\cdots,fc_{t+1,N}\end{aligned} \tag{4}$$

$\Psi_{t+1}(f)$ 表示在时刻 $t+1$,隐含状态为 f 时,概率最大的路径的第 t 个节点.计算公式如下:

$$\Psi_{t+1}(f)=arg\ \max(\delta_t(j)A_{jf}),$$
$$j=fc_{t,1},fc_{t,2},\cdots\cdots,fc_{t,N},$$
$$f=fc_{t+1,1},fc_{t+1,2},\cdots\cdots,fc_{t+1,N} \tag{5}$$

根据上述公式,维特比算法确定最大概率路径的过程如下:

(1)计算 $t=1$ 时刻的局部状态:

$$\delta_1(f)=\pi_f B_{fF_1} \tag{6}$$
$$\Psi_1(f)=0 \tag{7}$$

(2)动态规划递推 $t=2,3,\cdots\cdots,n$ 时刻的局部状态:

$$\delta_t(f)=\max(\delta_{t-1}(j)A_{jf}B_{fF_t}) \tag{8}$$
$$\Psi_t(f)=arg\ \max(\delta_{t-1}(j)A_{jf}) \tag{9}$$

(3)计算 $t=n$ 时刻最大的 $\delta_n(f)$ 和 $\Psi_n(f)$,即为 $t=n$ 时刻最大的 fc 序列概率取值及 fc:

$$P_{max}=\max(\delta_n(f)) \tag{10}$$
$$f_{n_{max}}=arg\max(\delta_n(f)) \tag{11}$$

(4)递推 fc 序列,对于 $t=n-1,n-2,\cdots\cdots,1$:

$$f_{t_{max}}=\Psi_{t+1}(f_{t+1_{max}}) \tag{12}$$

最终得到最有可能的 fc 序列$\{f_1,f_2,\cdots\cdots,f_n\}$.其中本文规定的初始状态概率矩阵 $\boldsymbol{\pi}$,隐含状态转移概率矩阵 $\boldsymbol{A}$ 及观测状态转移概率矩阵 $\boldsymbol{B}$ 的取值如表 3、表 4 所示.

其中,表 3 基于 4.3.2 节确定的离群值边界,初步参考对应的修正数据比例作为初始状态概率矩阵及观测状态转移概率矩阵的概率取值.然后,为了确保概率矩阵可以使用于更为复杂的网络环境中,本文将离群值边界扩展到 2 倍以及 4 倍以便涵盖更多的数据.随着边界的扩大,原有的数据中被视为离群值的数据点可能被重新归类为正常值.同时,由于边界的扩大可能会引入更多的噪声,本文相应地调整了概率矩阵的取值,以保证模型的泛化能力和适应性,从而使模型在不同的环境中仍然有效.

为了得到隐含状态转移概率矩阵,本文在 YouTube 平台模拟真实场景中的用户播放行为,得到以下四种视频片段跳转情况的比例:

(1)连续的相同分辨率视频片段跳转;

(2)非连续的相同分辨率视频片段跳转;

(3)连续的不同分辨率视频片段跳转;

(4)非连续的不同分辨率视频片段跳转.

此外,根据古德-图灵估计,对于出现次数非常少的情况,把一部分看得见的事件的概率匀给未看见的事件.本文为播放时跳转到不同视频的四种未现事件各分配了小概率 0.1%,最终得到概率矩阵如表 4 所示.

表 3　初始状态概率矩阵 π 及观测状态转移概率矩阵 B 设定

CAVCL 取值范围		概率
$MINUS_{flag}=0$	$MINUS_{flag}=1$	
$[CAVCL-10,CAVCL+10]$	$[CAVCL-133,CAVCL+145]$	0.80
$[CAVCL-20,CAVCL+20]$	$[CAVCL-266,CAVCL+290]$	0.15
$[CAVCL-40,CAVCL+40]$	$[CAVCL-532,CAVCL+580]$	0.05

表 4　隐含状态转移概率矩阵 A 设定

视频 ID		分辨率		是否为连续片段	概率
相同	不同	相同	不同		
√	×	√	×	√	0.800
				×	0.156
×	√	√	×	√	0.001
				×	0.001
√	×	×	√	√	0.030
				×	0.010
×	√	×	√	√	0.001
				×	0.001

一个视频 CAVCL 往往会匹配产生多个结果,若在寻找最大可能的 fc 序列时将所有匹配结果都纳入搜索范围,在视频识别时将耗费大量时间,因此,本文采取快速匹配方法.将可获取到的 CAVCLS 的前 σ 个 CAVCL 进行全部匹配,将得到所有 fc 构成匹配结果库,若在其匹配结果库中没有出现该视频 ID 对应的连续明文指纹组合,则放弃该视频 ID,去除匹配结果库中的无用视频明文,其余 CAVCL 在该局部明文指纹库其中进行匹配.最后,更新

CAVCLS 的匹配结果以计算最优路径，从而获得视频内容识别结果.

5 实验与分析

5.1 实验环境

本文使用自动化采集方法采集了加密视频流和视频明文指纹库，采集环境如图 14 所示.

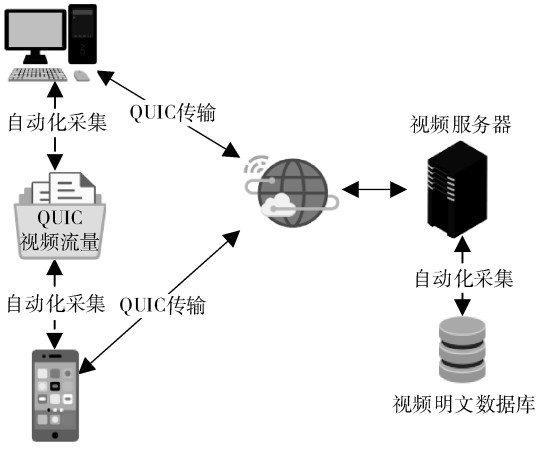

图 14 数据采集环境

视频播放数据采集的工作在美国洛杉矶进行. 对于 HTTP/3 加密视频流，设备通过 Wi-Fi 连接到 Internet，播放视频的同时抓取视频传输流量包. 由于需要大量 HTTP/3 视频的传输流量，本文使用自动化工具 UIBOT[33] 采集流量包：

(1)使用自动化程序在视频网站获取大量视频标题及对应网址并储存.

(2)UIBOT 操纵设备依次打开视频网址播放视频并记录对应分辨率及视频 ID. 对于 PC(Personal Computer)，使用 Wireshark 抓取视频流量；对于手机，使用 tcpdump 抓取视频流量.

对于 YouTube 视频明文数据，从服务器发送的响应 JSON(JavaScript Object Notation)文件中可以获取音视频索引信息. 由于只有在客户端获得索引信息的情况下，才能实现 DASH 技术的动态自适应播放功能，因此索引信息可以稳定获得. 为了快速获取索引信息，本文编写了自动化程序进行批量采集，具体方法如下：

(1)在视频网站爬取使用不同关键字搜索到的视频 JSON 文件.

(2)在 JSON 文件中提取索引片段在整个音视频中的所在范围，并根据该范围下载音视频索引片段.

(3)从片段中提取索引信息以获得音视频明文指纹信息，从而建立视频明文指纹数据库.

本文采用的明文指纹库构建方法成本较低，便于推广和扩展. 由于音视频索引片段的数据量较小，且使用了自动化采集程序实现指纹库的构建，因此所需的存储空间以及耗费的时间都较少. 本文采集的 YouTube 视频的大型明文指纹库中包含约 36 万条明文指纹序列，只需要 347MB 的存储空间，且在一天内即可完成. 另外使用类似方法构建 Facebook 和 Instagram 指纹库.

5.2 实验数据

5.2.1 明文指纹数据库

已有的使用密文指纹库的视频识别研究[5,6,22]多基于小型视频指纹库进行方法验证. 指纹库较小时，发生误判的可能性要远小于在大型数据库的可能性. 但是，在主干网监管的场景下，对视频识别需要在大型的指纹库中进行，虽然已有基于明文指纹库的方法[4,10]可以做到在大型指纹库中的识别达到较好的效果，但是其仅针对使用 HTTP/1.1 传输的 DASH 视频. 目前，没有方法说明其可以在十万级以上的指纹库中识别使用 HTTP/3 传输的 DASH 视频. 综上所述，使用大型指纹库可以真实地反映本方法的实用性.

本文分别采集构建了小型和大型明文指纹库. 其中，小型 YouTube 明文指纹库包含 6000 条真实指纹序列，小型 Facebook 明文指纹库包含 2111 条真实指纹序列. 为了验证本文方法在大型指纹库中的可用性，本文分别采集了 362502 条 YouTube 明文指纹序列和 283895 条 Facebook 明文指纹序列构建了大型 YouTube 和 Facebook 明文指纹库. 除此之外，本文构建了包含 74342 个明文指纹序列的 Instagram 大型指纹库用于方法通用性的验证. 目前为止，本文构建的视频指纹库是已知研究中规模最大的.

5.2.2 加密视频流量

为了验证本文方法的可行性，本文播放了 YouTube、Facebook 和 Instagram 三个视频平台上的视频. 具体数据如下：

(1)YouTube：播放了 2850 次视频，时长在 1 分钟至 10 分钟之间；

(2)Facebook：播放了 390 次视频，时长在 1 分钟至 10 分钟之间；

(3)Instagram：播放了 243 次视频，时长在 1 分钟至 2 分钟之间.

以上总计播放了 3483 次视频，时长总长约

16565 分钟.对于视频播放时传输的加密传输数据,本文提取其中的修正音视频片段组合长度作为待识别的 CAVCL,对 YouTube 平台共提取了 84715 个 CAVCL,对 Facebook 平台共提取了 18741 个 CAVCL,对 Instagram 平台,共提取了 6470 个 CAVCL.

本文使用的实验数据总体描述见表 5.

表 5　视频数据来源与数量

视频明文指纹库	明文指纹库中的指纹数量	播放的加密视频数量	提取的 CAVCL 数量
YouTube 小型指纹库	6000	2850	84715
YouTube 大型指纹库	362502		
Facebook 小型指纹库	2111	390	18741
Facebook 大型指纹库	283895		
Instagram 大型指纹库	74342	243	6470

5.3　结果评估测度

本文使用准确率 *Accuracy*、精确率 *Precision*、召回率 *Recall* 以及 *F*1 得分 *F*1_*score* 四个测度对识别结果进行了评估. *Accuracy*、*Precision*、*Recall* 的计算依赖于四种结果 *TP*(True Positive)、*TN*(True Negative)、*FP*(False Positive)、*FN*(False Negative).在本文中,对于精确率和召回率,每个类别都需要单独计算,并使用加权平均方法计算总精确率和总召回率,总准确率被定义为分类正确的样本数与总样本数的比值.

对于每个类别,测度的计算公式如下:

$$precision = \frac{TP}{TP + FP} \tag{13}$$

$$recall = \frac{TP}{TP + FN} \tag{14}$$

$$F1\_score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{15}$$

准确率、精确率和召回率需要联合使用才能全面评估算法的可用性.F1 得分被定义为精确率和召回率的调和平均数,同时兼顾了分类的精确率和召回率.

5.4　实验结果

本文首先验证了音视频长度序列作为视频指纹可以唯一确定视频内容的可行性,其次,对本文方法分别在各视频平台小型指纹库及大型指纹库中进行了实验验证,最后,将本文方法与类似的工作进行比较分析.在实验中,对于快速匹配方法中 CAVCLS 需进行全部匹配的前 σ 个 CAVCL,本文设置了不同的 σ 取值,并在不同的匹配阈值下进行识别,对得到的准确率、精确率、召回率、F1 得分四个测度进行评估.

5.4.1　音视频长度序列唯一性验证

本文方法使用的视频指纹为音视频长度序列,使用音视频长度序列可唯一确定视频内容是本文视频识别的基础.以下将对音视频长度序列的唯一性进行验证.

当不同的视频具有相同的音视频长度序列的可能性接近 0,就可以认为使用音频长度序列可以唯一确定视频的内容.本文使用实际采集的 YouTube 大型指纹库中明文指纹相等的概率,来计算不同视频音视频长度序列相同的概率.

本文选取包含游戏、音乐、影视、科技等方面的多个关键词,并利用这些关键词随机选取部分视频构建视频明文指纹库.根据统计学知识,对整体进行随机抽样,当样本容量足够大的时候,就可以保证样本具有代表性和独立性,从而能够准确反映总体的特征.明文指纹库中包含 362502 个视频的约 34996035 个音视频片段,这些片段长度的概率密度函数(Probability Density Function, PDF)如图 15 所示.从图 15 可知,任意两个音视频片段相等的概率最高不超过 1×10^{-6}.

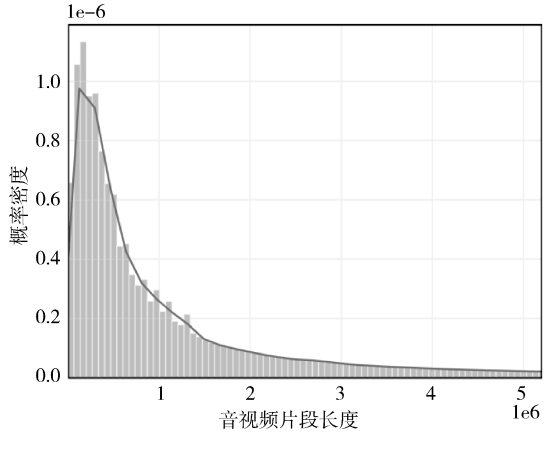

图 15　YouTube 数据集音视频片段长度的概率密度函数

两个音视频片段长度序列相同是指组成序列的片段长度都相等,这个概率为对应位置的两个片段长度相等的概率之积.图 16 展示了从 YouTube 平台的加密视频流量数据提取出的 CAVCLS 中包含的 CAVCL 数量的概率分布,视频的时长大多在 1

分钟至 10 分钟之间.可以看出,CAVCLS 中包含的 CAVCL 数量在 20 个至 30 个之间,其中,CAVCL 中一般包含 1 至 8 个音视频片段.即使取分段长度相同的最高概率 1×10^{-6},且假设 CAVCL 中仅包含 1 个音视频片段,对于包含 20 个 CAVCL 的视频来说,视频之间指纹序列完全相同的概率也仅为 $(1\times10^{-6})^{20}$.对于时长更长的视频,其中包含的音视频片段只会更多,因此视频指纹序列相同的概率会更低,接近于 0.

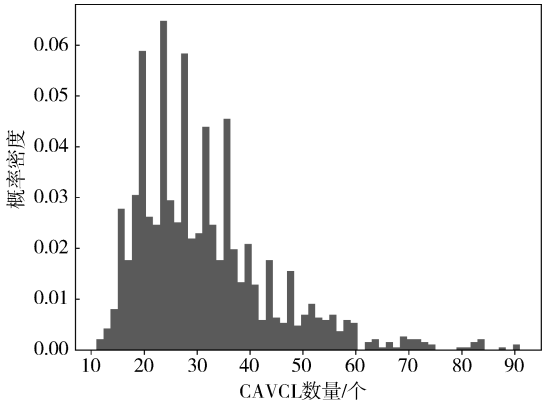

图 16 YouTube 平台 CAVCLS 中包含的 CAVCL 数量

根据上述计算结果,不同视频具有相同音视频长度序列长度的可能性接近 0,因此,使用音视频长度序列作为指纹唯一确定视频内容是可行的.

5.4.2 小型指纹库中的实验结果与分析

为了降低全部匹配带来的过长识别时间,本文采用了快速匹配方法.将可获取到的 CAVCLS 的前 σ 个 CAVCL 进行全部匹配,并将剩余的 CAVCL 进行局部匹配.本文首先需要确定 σ 的大致取值,因此需要对单个视频所包含的 CAVCL 数量分布进行统计.

从图 16 中可以看出大部分视频传输所产生的 CAVCL 数量在 20 个至 30 个之间,因此,对于 YouTube,本文根据 CAVCL 数量的三分之一至二分之一大小分别设置 σ 的取值为 10、11、12、13、14、15 进行实验,并与对所有 CAVCLS 使用全匹配的识别结果作为对比.

本文首先在包含真实指纹的小型指纹库中进行实验验证.对于 YouTube 平台,在 σ 分别取 10、11、12、13、14、15 的情况下,设置最小匹配区间 MI (Matching Interval)分别为

(1) MI_1 ($MINUS_{flag}=0$:$[CAVCL-10, CAVCL+10]$, $MINUS_{flag}=1$:$[CAVCL-133, CAVCL+145]$);

(2) MI_2 ($MINUS_{flag}=0$:$[CAVCL-20, CAVCL+20]$, $MINUS_{flag}=1$:$[CAVCL-266, CAVCL+290]$);

(3) MI_3 ($MINUS_{flag}=0$:$[CAVCL-40, CAVCL+40]$, $MINUS_{flag}=1$:$[CAVCL-532, CAVCL+580]$).

使用不同的 CAVCL 数量,以及设置了不同的匹配范围后,在小型的 YouTube 指纹库中进行视频识别,结果如表 6 所示.

表 6 YouTube 小型指纹库中不同 MI 和 σ 对应的结果

σ	匹配范围	准确率	精确率	召回率	F1 得分
10	MI_1	0.9475	0.9795	0.9475	0.9474
	MI_2	0.9583	0.9862	0.9583	0.9595
	MI_3	**0.9613**	**0.9872**	**0.9613**	**0.9627**
11	MI_1	0.9699	0.9880	0.9699	0.9696
	MI_2	**0.9743**	**0.9919**	**0.9743**	**0.9761**
	MI_3	0.9725	0.9913	0.9725	0.9750
12	MI_1	0.9803	0.9915	0.9803	0.9809
	MI_2	**0.9818**	**0.9936**	**0.9818**	**0.9836**
	MI_3	0.9788	0.9914	0.9788	0.9806
13	MI_1	**0.9862**	**0.9948**	**0.9862**	**0.9873**
	MI_2	0.9855	0.9947	0.9855	0.9869
	MI_3	0.9821	0.9936	0.9821	0.9844
14	MI_1	0.9885	0.9955	0.9885	0.9898
	MI_2	**0.9881**	**0.9959**	**0.9881**	**0.9899**
	MI_3	0.9844	0.9946	0.9844	0.9870
15	MI_1	0.9929	0.9969	0.9929	0.9941
	MI_2	**0.9918**	**0.9981**	**0.9918**	**0.9943**
	MI_3	0.9877	0.9954	0.9877	0.9905
全部匹配	MI_1	**0.9963**	**0.9991**	**0.9963**	**0.9975**
	MI_2	0.9937	0.9994	0.9937	0.9962
	MI_3	0.9888	0.9961	0.9888	0.9915

表 6 展示了 YouTube 对应的小型明文指纹库中的视频识别结果.使用 F1 得分作为识别结果好坏的主要标准,从表 6 中可以看出,不同的 σ 取值下所取得最好结果的匹配范围不唯一,大多数情况下会在范围为 MI_2 时取得最佳的综合结果;对于同一个匹配范围,σ 的取值越大 F1 得分越高,这是因为当播放分辨率固定时,匹配的 CAVCL 越多,识别结果越准.使用快速匹配的方法所得到的综合效果中最好的 F1 得分仅比对 CAVCLS 进行全匹配低 0.32%,且精确率达到了 99.81%,这说明仅对 CAVCLS 进行部分的全部匹配,也可以得到非常准确的结果.

表 6 中最好的快速匹配时的 F1 得分是在 σ 为 15,匹配范围为 MI_2 时取得的,为 99.43%,此时视

频识别的准确率、召回率和精确率都超过了 99%.

与 YouTube 平台类似,对于 Facebook 采用同样的原理设置实验.分别设置 σ 的取值为 15、16、17、18 进行实验,并取对所有 CAVCLS 使用全匹配的识别结果作为对比.由于 Facebook 平台不存在 $MINUS_{flag}=1$ 的情况,因此设置最小匹配区间分别为:

(1)$MI_1([CAVCL-180, CAVCL+179])$;

(2)$MI_2([CAVCL-360, CAVCL+358])$;

(3)$MI_3([CAVCL-720, CAVCL+716])$.

Facebook 为了混淆音视频片段组合在传输过程中的大小信息,在流量中采用了填充帧,导致 Facebook 平台的密文长度还原具有更强的不确定性,因此其匹配区间较 YouTube 更大.

考虑到篇幅限制,后续针对不同平台的实验结果只展示每个 σ 值在三种匹配范围中获取最高 F1 得分的一项作为最佳识别结果.

表 7 展示了对于不同的 σ,使用 $MI_1 \sim MI_3$ 这三个匹配范围在小型 Facebook 明文指纹库中进行测试后,所得到的最佳匹配范围及对应的识别结果.可以看到,由于加密流量中填充帧的存在,整体的识别综合结果较 YouTube 平台略低.在 σ 为 17,匹配范围为 MI_1 时,快速匹配整体 F1 得分最高为 97.29%,仅比全匹配低 0.15%.

表 7　Facebook 小型指纹库中不同 σ 对应的最佳识别结果

σ	匹配范围	准确率	精确率	召回率	F1 得分
15	MI_2	0.9605	0.9921	0.9605	0.9683
16	MI_2	0.9632	0.9921	0.9632	0.9699
17	MI_1	0.9658	0.9949	0.9658	0.9729
18	MI_1	0.9658	0.9949	0.9658	0.9729
全部	MI_1	0.9684	0.9949	0.9684	0.9744

总体看来,本方法在小型指纹库中具有较高的识别精确率.虽然全匹配下的视频识别各项评估测度较高,但是快速匹配降低了匹配时间,可以在对较少的 CAVCL 进行全部匹配的情况下,实现与全匹配各项测度相差不到 0.5%的识别效果.对于存在加密填充的平台,由于每次填充长度的不同,导致还原误差增大,进而导致识别准确率降低.

5.4.3　大型指纹库中的实验结果与分析

为了进一步说明本方法在实际中的可行性,对于 YouTube、Facebook 和 Instagram 平台,本文在大型指纹库中也进行了实验验证.

对于 YouTube 平台,不同的 σ 取值在 $MI_1 \sim MI_3$ 这三种匹配范围中,能够得到最佳 F1 得分的匹配范围及对应的识别结果如表 8 所示.随着 σ 取值的增大,F1 得分逐渐增加,与使用 CAVCLS 全匹配相比,当 σ 取 15 时就已经可以取得与其相似的结果.

表 9 展示了对于不同的 σ,使用 $MI_1 \sim MI_3$ 这三个匹配范围在大型 Facebook 明文指纹库中进行测试后,所得到的最佳匹配范围及对应的识别结果.由表 9 可知,当 σ 取 18 时就已经可以达到与使用 CAVCLS 全匹配相近的结果.

表 8　YouTube 大型指纹库中不同 σ 对应的最佳识别结果

σ	匹配范围	准确率	精确率	召回率	F1 得分
10	MI_2	0.9435	1.0000	0.9435	0.9581
11	MI_1	0.9613	0.9997	0.9613	0.9711
12	MI_1	0.9717	0.9997	0.9717	0.9804
13	MI_1	0.9773	0.9997	0.9773	0.9850
14	MI_1	0.9799	0.9997	0.9799	0.9873
15	MI_1	0.9840	0.9997	0.9840	0.9906
全部	MI_1	0.9877	1.0000	0.9877	0.9932

表 9　Facebook 大型指纹库中不同 σ 对应的最佳识别结果

σ	匹配范围	准确率	精确率	召回率	F1 得分
15	MI_1	0.9368	1.0000	0.9368	0.9558
16	MI_1	0.9395	1.0000	0.9395	0.9574
17	MI_1	0.9447	1.0000	0.9447	0.9613
18	MI_1	0.9474	1.0000	0.9474	0.9631
全部	MI_1	0.9500	1.0000	0.9500	0.9645

为了更全面地验证本方法的通用性,对于 Instagram 平台,采用与论文 5.4.2 节同样的原理设置实验.分别设置 σ 的取值为 12、14、16、18 进行实验,并取对所有 CAVCLS 使用全匹配的识别结果作为对比.由于 Instagram 平台不存在 $MINUS_{flag}=1$ 的情况,因此设置最小匹配区间分别为:

(1)$MI_1([CAVCL-148, CAVCL+143])$;

(2)$MI_2([CAVCL-296, CAVCL+286])$;

(3)$MI_3([CAVCL-592, CAVCL+572])$.

与 Facebook 平台类似,Instagram 平台同样采用了填充帧,导致密文长度还原具有不确定性,匹配区间也较大.表 10 展示了对于不同的 σ,使用不同的匹配范围在大型 Instagram 明文指纹库中进行测试后,所得到的最佳匹配范围及对应的识别结果.由于加密流量中填充帧的存在,整体的识别综合结果较 YouTube 平台略低.在 σ 为 18,匹配范围为 MI_1 时,快速匹配整体 F1 得分最高为 97.35%,仅比全匹配低 0.22%.

表 10 Instagram 指纹库中不同 σ 对应的最佳识别结果

σ	匹配范围	准确率	精确率	召回率	F1 得分
12	MI_1	0.9342	1.0000	0.9342	0.9631
14	MI_1	0.9383	1.0000	0.9383	0.9661
16	MI_1	0.9465	1.0000	0.9465	0.9710
18	MI_1	0.9506	1.0000	0.9506	0.9735
全部	MI_1	0.9547	1.0000	0.9547	0.9757

总的来看，YouTube、Facebook、Instagram 在大型指纹库中的最佳识别结果对应的匹配范围大多都是 MI_1，这是因为较小的匹配范围可以在大型指纹库中有效地将错误匹配结果剔除出去.

可以看出，总体上大型指纹库上的最佳 F1 得分结果要略差于小型指纹库上的结果. 小型指纹库中匹配的召回率总是大于大型指纹库，精确率小于大型指纹库，这是因为随着指纹库规模的增加，视频 CAVCLS 被错误识别为其他不在样本中的视频的明文指纹的概率增加，FN 事件也会随之增加，反而 FP 事件减少，因此在小型指纹库中适用的识别参数在大型指纹库中未必适用. 但从整体来看，本文的视频识别方法在大型指纹库上的结果和小型指纹库上的结果相差不大，这也说明了本文方法具有实用性.

5.4.4 不同规模指纹库下所需的视频识别时间

表 11 展示了不同规模的指纹库下 Facebook 平台在同一 σ、不同匹配区间下进行视频识别的时间消耗. 实验采用的具体配置为：CPU 为 Intel i9-10900k；操作系统为 Windows 10；内存容量为 128GB. 从表中可以看出，对于同一个匹配范围，不同的 σ 取值需要消耗的时间相差并不大. 根据表 6～表 10 中的匹配结果可知，F1 得分在 σ 取值增大到一定数值时就已经表现出与全匹配相差不到 0.5% 的效果. 对于本文提出的方法，只需要根据平台的视频传输模式选择合适的 σ 取值即可达到较好的识别效果.

表 11 Facebook 平台小型及大型指纹库下识别时间消耗

σ	匹配范围	小型指纹库			大型指纹库		
		时间 (s)	时间 (s)/个	F1 得分	时间 (s)	时间 (s)/个	F1 得分
15	MI_1	1.747	0.004	0.9662	156.680	0.338	0.9558
	MI_2	3.569	0.008	0.9683	629.646	1.357	0.9489
	MI_3	9.955	0.021	0.9631	2197.112	4.735	0.9412
16	MI_1	1.818	0.004	0.9678	162.232	0.350	0.9574
	MI_2	3.442	0.007	0.9699	617.697	1.331	0.9465
	MI_3	10.094	0.022	0.9631	2301.085	4.959	0.9397
17	MI_1	1.809	0.004	0.9729	165.634	0.357	0.9613
	MI_2	3.489	0.008	0.9726	637.402	1.374	0.9465
	MI_3	10.285	0.022	0.9647	2329.900	5.021	0.9412
18	MI_1	1.840	0.004	0.9729	168.821	0.364	0.9631
	MI_2	3.447	0.007	0.9726	666.885	1.437	0.9520
	MI_3	10.458	0.023	0.9647	2417.746	5.211	0.9412

对于同一个 σ，在小型指纹库中，即使是较大的匹配范围，每个视频也只需要不到 0.03s 的时间进行匹配. 在大型指纹库中，根据表 8～表 10 中的匹配结果可知，并不是匹配范围越大越准确. 总的来看，YouTube、Facebook 和 Instagram 在大型指纹库中的最佳识别结果对应的匹配范围大多都是 MI_1. 对于本文提出的方法，只需要使用识别时间较短的 MI_1，就可以达到较好的识别结果.

因此，对于本文提出的方法，只需要选择较小的识别区间，并根据平台选择合适的 σ 取值，就可以达到识别时间短与效果好的平衡.

5.4.5 与类似工作的比较分析

HTTP/3 虽然已经在 YouTube、Facebook 和 Instagram 等头部视频分享网站广泛部署，但是该协议的标准 2022 年才发布，现有研究中也没有针对 HTTP/3 的视频识别研究. 本文与类似工作的对比分别面向本文 4.3 和 4.4 的关键方法.

(1)密文的长度修正方法

对密文长度的准确修正是加密视频识别的基础. 4.3 节提出了密文的长度修正算法，与本文类似，Xu 等人在研究 HTTPS/QUIC 视频的自适应切换行为时[27]，也对 QUIC 流量进行了分块和长度修正处理. 该研究将密文块的长度直接估计为“去掉 IP/UDP/QUIC 头部的所有数据报中的 QUIC 载荷长度之和”，并没有对密文块长进行进一步的修正，最终得到 QUIC 块的大小误差在 5%，相当于本文原始误差率的两倍. 由于其目标是识别分辨率切换行为，一次仅需要对一个视频 ID 的不同分辨率进行识别，因此 5% 的误差不会影响分辨率识别的准确性. 本文将该论文的误差范围用于视频识别，在不同规模的 YouTube 指纹库上的评估结果如表 12 所示.

表 12 Xu[27]方法的长度修正误差在不同规模 YouTube 指纹库中的视频识别结果

指纹库大小	准确率	精确率	召回率	F1 得分
6000	0.0952	0.8707	0.0952	0.0812
362502	0	0	0	0

表 12 的结果表明，使用文献[27]的误差修正范围只能在很小的视频指纹库中进行识别，在大型指纹库中，即使使用全部 CAVCL 进行全匹配也无法准确识别任一视频，这表示其方法无法应用于大规

模的视频识别上.

(2)基于指纹库进行视频识别

4.4 节给出了将修正后的长度特征在指纹库中进行视频识别的方法,虽然文献[10]也使用了基于明文指纹库的类似思路在大型指纹库中进行视频识别,但是该论文需要从 TCP 的头部提取特征,因此无法应用于使用 UDP 传输的 HTTP/3 协议,也就无法用相同的数据集进行对比.

与本文的方法不同的是,基于密文指纹库的方法结合深度学习,可以应用于多个版本的 HTTP 协议,因此本文选择其中代表性的成果进行对比. Bae 等人[22]在 2022 年研究了 LTE 网络中的视频识别,在 LTE 网络中,无特权的攻击者可以监控受害者的下行链路流量,从而识别正在观看目标视频的移动用户,然后推断这些用户的每一个正在观看的视频标题,并实现了高达 98.5%的准确率.

Bae 方法基于密文指纹库对加密视频进行识别. 对于每个需要识别的目标视频,该方法需要每个视频超过 27 次的播放数据. 在获取足够的密文数据后,将每 0.2 秒的聚合流量大小序列作为 CNN 模型输入对视频进行标题分类.

本文使用 5.1 节中的自动化采集方法收集了 YouTube 平台的 20 个视频的 37 遍播放数据进行实验. 每个视频的播放过程中采取固定分辨率的播放模式,时长为 2 至 3 分钟之间. 由于 Bae 方法基于密文指纹库,本文方法基于明文指纹库,因此分两种情况进行实验,对于 Bae 方法,将训练集的数量分别设置为 7、12、17、22 和 27,对于本文的方法,分别设置大小为 20、6000、362502 的明文指纹库. 对于以上两种方法,使用同样的 10 遍播放数据进行验证.

实验结果如表 13、表 14 所示. 从表 13 中可以看出,Bae 方法在训练数据减少时,准确率、精确率、召回率、F1 得分都有所下降. 在采集数据时,需要对完整的视频进行播放,在不间断采集数据的情况下,20 个 2 至 3 分钟视频的 27 遍播放数据大约需要 1080 至 1620 分钟. 因此对于更大规模的视频识别,Bae 方法需要大量的密文数据作为支撑,这就需要更多的时间进行密文指纹库的构建. 因此使用密文指纹的视频识别方法通常只能适用规模很小的视频库.

表 14 为本文方法在 YouTube 各规模指纹库中的实验结果. 在较小的指纹库中,本文方法的各项评估测度都达到了 97%以上,在大型指纹库中,达到了 95%以上,其中精确率为 100%. 而且本文方法仅需视频的元信息来提取明文指纹,在不到一天(1440 分钟)的时间内就可以构建 36 万级别的明文指纹库,且明文指纹是视频稳定的特征,不会随着采集环境的网络状况变化而变化,相较密文指纹更加可靠.

表 13　Bae 方法在不同训练集大小下的实验结果

训练集大小	准确率	精确率	召回率	F1 得分
27	0.9750	0.9847	0.9650	0.9748
22	0.9700	0.9897	0.9600	0.9701
17	0.9450	0.9741	0.9400	0.9567
12	0.9350	0.9531	0.9150	0.9347
7	0.9200	0.9574	0.9000	0.9196

表 14　本文方法在不同指纹库大小下的实验结果

指纹库大小	准确率	精确率	召回率	F1 得分
20	1	1	1	1
6000	0.9750	1	0.9750	0.9860
362502	0.9500	1	0.9500	0.9678

本文方法基于领域知识提取包括数据传输和控制信息两种特征,可以快速地对视频进行识别,所需数据量少,而 Bae 方法所使用的深度学习所需的训练数据多,并且视频识别结果的优劣依赖于数据集的规模,性能和泛化能力较差. 因此,在真实应用场景中,本文方法相较使用深度学习和密文指纹库的方法更加实用.

6　成果应用分析

本文的研究成果将在网络空间安全领域发挥重要作用. 互联网的全球性和其路由策略的自治特性使得应用信息的传播跨越管理边界,这由互联网的体系结构所决定,也是当前网络管理难题的根源. 随着互联网与社会生活的紧密结合,信息来源的复杂多样性带来了新的安全挑战,尤其是有害视频内容的传播,迫切需要在现有互联网架构上开发有效的监管技术. 同时,保护用户隐私和支持跨自治域的信息自由分享也成为现代社会的必要需求. 本研究提供的技术方案,旨在实现这两个目标.

本文方法能够从网络接入点采集的流量中识别出特定的有害视频内容. 由于该方法基于网络接入点采集数据,无需视频平台的配合,克服了自治域管理权限的限制,使得实时监管本自治域内的有害视频成为可能. 此技术支持从加密流量中识别有害视频,为监管部门提供及时的细粒度管控依据. 该方法

依赖于大规模的有害视频指纹库,该库需由管理部门认定并更新,不涉及库外视频的监管,保证了正常网络使用的自由和隐私.尽管 HTTP/3 在国内尚未广泛应用,本研究的数据采集主要针对国外视频平台,但随着 HTTP/3 的优势日益凸显,预期将成为未来的发展方向,本研究方法也将为国内采用 HTTP/3 的视频平台提供监管支持.

通过本文的研究成果,网络管理者将具有精细化的网络监管能力,能够针对网络中传输的视频内容采取不同的管理策略.本文方法的应用可在不改变现有网络架构的前提下同时解决网络公害监管和公民隐私保护的问题,具有很强的应用价值.

7 结束语

HTTP/3 使用了 QUIC 作为传输层协议对视频进行加密传输,这给视频识别带来了新的挑战.为了解决 HTTP/3 协议在视频网站中的应用导致现有加密识别方法失效的问题,本文提出了一种从加密 HTTP/3 流中识别出 DASH 视频内容的视频识别方法.首先,本文基于 HTTP/3 协议传输 DASH 视频时的特性,通过提取音视频片段组合传输层长度特征并进行修正得到应用层长度的近似值 CAVCLS,其误差率相较现有块长度修正方法大大降低.然后,利用 HMM 建立视频识别模型,将修正后的长度序列 CAVCLS 与视频明文指纹库生成的组合指纹库进行匹配以获取视频身份,并使用快速匹配方法降低视频匹配时间.为了证实本文方法在实际应用中的可行性,本文采集了 YouTube、Facebook 和 Instagram 视频平台的视频流量,分别在小型指纹库和大型指纹库中进行了实验验证.实验结果表明,本文的方法具有通用性,且在大型明文指纹库中,对 HTTP/3 加密视频的识别可以获得最高 99.4%的 F1 得分,是一种具有实际应用前景的 HTTP/3 视频识别方法.

HTTP/3 协议作为一种新型协议标准,在被逐步推广的同时,还在不断优化更新,未来应保持对该协议标准的关注,以及时地对本方法的特征选取、模型设计做出调整,使得本文方法可以适应复杂多变的网络环境.此外,未来国内视频平台也会逐步转向使用具有更高传输效率的 HTTP/3 协议,因此应当及时跟进国内视频平台的技术发展,并进行针对性研究,以满足国家对网络空间安全管理的要求.

参 考 文 献

[1] Ni B, Peng H, Chen M, et al. Expanding language-image pretrained models for general video recognition // Computer Vision-ECCV 2022: 17th European Conference. Tel Aviv, Israel, 2022: 1-18

[2] Kumar K, Shrimankar D D, Singh N. Eratosthenes sieve based key-frame extraction technique for event summarization in videos. Multimedia Tools and Applications, 2018, 77(6): 7383-7404

[3] Pu Zhan-Xing, Ge Yong-Xin. Few-shot action recognition in video based on multi-feature fusion. Chinese Journal of Computers, 2023, 46(3): 594-608(in Chinese)
(蒲瞻星, 葛永新. 基于多特征融合的小样本视频行为识别算法. 计算机学报, 2023, 46(3): 594-608)

[4] Reed A, Kranch M. Identifying HTTPS-protected Netflix videos in real-time // Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. Scottsdale, USA, 2017: 361-368

[5] Schuster R, Shmatikov V, Tromer E. Beauty and the burst: Remote identification of encrypted video streams // Proceedings of the 26th USENIX Security Symposium (USENIX Security 17). Vancouver, Canada, 2017: 1357-1374

[6] Gu J, Wang J, Yu Z, et al. Walls have ears: Traffic-based side-channel attack in video streaming // Proceedings of the IEEE Conference on Computer Communications. Honolulu, USA, 2018: 1538-1546

[7] Yang L, Fu S, Luo Y, et al. Markov probability fingerprints: A method for identifying encrypted video traffic // Proceedings of the 2020 16th International Conference on Mobility, Sensing and Networking (MSN). Tokyo, Japan, 2020: 283-290

[8] Stikkelorum M. I Know What You Watched: Fingerprint Attack on YouTube Video Streams // Proceedings of the 27th Twente Student Conference on IT. Enschede, Netherlands, 2017

[9] Reed A, Klimkowski B. Leaky streams: Identifying variable bitrate DASH videos streamed over encrypted 802.11n connections // Proceedings of the 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC). Las Vegas, USA, 2016: 1107-1112

[10] Wu Hua, Yu Zhen-Hua, Cheng Guang, et al. Encrypted video recognition in large-scale fingerprint database. Journal of Software, 2021, 32(10): 3310-3330(in Chinese)
(吴桦, 于振华, 程光等. 大型指纹库场景中加密视频识别方法. 软件学报, 2021, 32(10): 3310-3330)

[11] Iyengar J, Thomson M. RFC 9000: QUIC: A UDP-based mult-iplexed and secure transport, 2021-5, https://datatracker.ietf.org/doc/html/rfc9000

[12] Bishop M. RFC 9114: HTTP/3, 2022-6, https://datatracker.ietf.org/doc/html/rfc9114

[13] Nazir A, Khan R A. A novel combinatorial optimization based feature selection method for network intrusion detection. Computers & Security, 2021, 102: 102164

[14] Horowicz E, Shapira T, Shavitt Y. A few shots traffic classification with mini-FlowPic augmentations// Proceedings of the 22nd ACM Internet Measurement Conference. Nice, France, 2022: 647-654

[15] Herley C. Automated detection of automated traffic // Proceedings of the 31st USENIX Security Symposium (USENIX Security 22). Boston, USA, 2022: 1615-1632

[16] Zhang S, Wang Z, Yang J, et al. MineHunter: A practical cryptomining traffic detection algorithm based on time series tracking// Proceedings of the Annual Computer Security Applications Conference. Virtual Event, USA, 2021: 1051-1063

[17] Malekghaini N, Akbari E, Salahuddin M A, et al. Deep learning for encrypted traffic classification in the face of data drift: An empirical study. Computer Networks, 2023, 225: 109648

[18] Diallo A F, Patras P. Adaptive clustering-based malicious traffic classification at the network edge// Proceedings of the IEEE Conference on Computer Communications. Vancouver, Canada, 2021: 1-10

[19] Li Y, Huang Y, Xu R, et al. Deep content: Unveiling video streaming content from encrypted WiFi traffic // Proceedings of the Network Computing and Applications. Cambridge, USA, 2018: 1-8

[20] Li Y, Huang Y, Seneviratne S, et al. From traffic classes to content: A hierarchical approach for encrypted traffic classification. Computer Networks, 2022, 212: 109017

[21] Gu J, Wang J, Yu Z, et al. Traffic-based side-channel attack in video streaming. IEEE/ACM Transactions on Networking, 2019, 27(3): 972-985

[22] Bae S, Son M, Kim D, et al. Watching the watchers: Practical video identification attack in LTE networks// Proceedings of the 31st USENIX Security Symposium (USENIX Security 22). Boston, USA, 2022: 1307-1324

[23] Zhang X, Xiong G, Li Z, Yang C, et al. Traffic spills the beans: A robust video identification attack against YouTube. Computers & Security, 2024, 137:103623

[24] Feng L, Chung J W, Claypool M. Silhouette: Identifying YouTube video flows from encrypted traffic// Proceedings of the the 28th ACM SIGMM Workshop. Amsterdam, Netherlands, 2018: 19-24

[25] Amjad F, Khan F, Tahir S, et al. ENCVIDC: an innovative approach for encoded video content classification. Neural Computing and Applications, 2022, 34(21): 18685-18702

[26] Mazhar M H, Shafiq Z. Real-time video quality of experience monitoring for HTTPS and QUIC// Proceedings of the IEEE Conference on Computer Communications. 2018:1331-1339

[27] Xu S, Sen S, Mao Z M. CSI: inferring mobile ABR video adaptation behavior under HTTPS and QUIC // Proceedings of the Fifteenth European Conference on Computer Systems. Heraklion, Greece, 2020: 1-16

[28] Wu H, Cheng G, Hu X. Inferring ADU combinations from encrypted QUIC stream// Proceedings of the 14th International Conference on Future Internet Technologies. Phuket, Thailand, 2019: 1-6

[29] Qin Y Y, Hao S, Pattipati K R, et al. ABR streaming of VBR-encoded videos: Characterization, challenges, and solutions // Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies. Heraklion, Greece, 2018: 366-378

[30] Sodagar I. The mpeg-dash standard for multimedia streaming over the Internet. IEEE Multimedia, 2011, 18(4): 62-67

[31] Pan Wu-Bin, Cheng Guang, Wu Hua, Xu Jian. A method for identifying the QoE parameter of encrypted YouTube traffic in mobile network. Chinese Journal of Computers, 2018, 41(11): 2437-2452(in Chinese)
(潘吴斌,程光,吴桦,徐健. 移动网络加密 YouTube 视频流 QoE 参数识别方法. 计算机学报, 2018, 41(11): 2437-2452)

[32] Viterbi A J, Omura J K. Principles of Digital Communication and Coding. Dover edition. New York, USA: Dover Publications, Incorporated, 2013.

[33] Laiye. "UIBOT", https://www.uibot.com.cn/.

WU Hua, Ph. D., associate professor. Her research interests include encrypted traffic analysis, cyberspace security, network situational awareness.

NI Shan-Shan, M. S. candidate. Her research interests include encrypted traffic analysis, encrypted video identification.

LUO Hao, M. S. candidate. His research interests include encrypted traffic analysis, encrypted video identification.

CHENG Guang, Ph. D., professor. His research interests include cyberspace security monitoring and protection, network traffic big data analysis, botnet and APT attack detection, etc.

HU Xiao-Yan, Ph. D., associate professor. Her research interests include encrypted traffic analysis, cyberspace security, future network architecture.

Background

The majority of video platforms on the Internet use end-to-end encryption to protect user privacy, as video applications have become the main source of business traffic. However, the application of encryption technology also provides the possibility for the dissemination of harmful videos. Therefore, rapid identification of harmful videos transmitted over the Internet is a necessary prerequisite for effective management of cyber space.

Existing methods have utilized the transmission characteristics of the HTTP/1.1 protocol and combined them with video service distribution mechanisms to recognize video content. However, as transmission and encryption protocols are constantly evolving, the widespread use of the HTTP/3 protocol has rendered previous methods ineffective. Moreover, few existing methods can demonstrate their practicality in large-scale fingerprint database scenarios.

This paper presents a novel approach to identifying encrypted videos transmitted using the HTTP/3 protocol in large-scale fingerprint databases, and provides a comprehensive evaluation of the approach using a database containing real video fingerprints. Experimental results demonstrate that the approach is highly practical.

This research is supported by the National Key R&D Program (No. 2021YFB3101403), which aims to provide technical support for effective management of cyber space. Our research group has been engaged in encrypted traffic analysis for many years and has conducted in-depth research on video transmission technologies. We have published several related papers, and our previously published research on encrypted video identification for HTTP/1.1 has been granted multiple invention patents. This paper is one of the core research achievements of our project, which aims to address the challenge of identifying encrypted videos in the context of the widely adopted HTTP/3 protocol.