软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(7):3375−3404 [doi: 10.13328/j.cnki.jos.007236] [CSTR: 32375.14.jos.007236]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向 HTTP/2 流量多路复用特征的加密视频识别方法*

吴 桦[1,2,3], 罗 浩[1], 赵士顺[1], 刘嵩涛[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, 江苏 南京 211189)

[2](网络通信与安全紫金山实验室, 江苏 南京 211111)

[3](网络空间国际治理研究基地 (东南大学), 江苏 南京 211189)

[4](江苏省泛在网络安全工程研究中心, 江苏 南京 211189)

通信作者: 吴桦, E-mail: hwu@seu.edu.cn

摘 要: 视频应用平台的兴起使得视频得以快速传播并渗透社会生活的各个方面. 网络中传播的视频也混杂了一些公害视频, 因此网络空间安全监管迫切需要准确地识别网络中加密传播的公害视频. 已有方法在网络主要接入点采集流量数据, 提取加密视频流量的特征, 基于公害视频库, 通过流量特征的匹配识别出被传输的公害视频. 然而随着视频加密传输协议的更新, 使用新型多路复用技术的 HTTP/2 协议已经大规模部署应用, 这导致传统的基于 HTTP/1.1 传输特征的流量分析方法无法识别使用 HTTP/2 传输的加密视频. 此外, 当前的研究大多针对的是播放时分辨率固定的视频, 很少考虑到流媒体自适应播放时分辨率切换给识别带来的影响. 针对以上问题, 详细分析了视频平台使用 HTTP/2 协议传输视频时音视频数据长度发生偏移的原理, 并提出了将多路复用的加密数据精准修正还原为组合音视频数据单元长度的方法, 从而构建出精准还原的加密视频修正指纹. 然后, 利用加密视频修正指纹和大型视频明文指纹库, 提出了视频修正指纹滑动匹配机制和以隐马尔可夫模型与维特比算法为基础的加密视频识别模型. 该模型使用动态规划方法解决了视频分辨率自适应切换带来的问题, 其在 40 万级的 Facebook 和 Instagram 真实指纹库场景中, 对固定分辨率和自适应分辨率的加密视频的识别准确率分别达到了 98.41% 和 97.91%. 使用 Triller、Twitter 和芒果 TV 这 3 个视频平台进行了方法通用性和泛化性验证. 与类似工作在识别效果、泛化性和时间开销方面的比较进一步验证了所提出的方法具有较高的应用价值.

关键词: HTTP/2; DASH; 大型数据集; 修正指纹; HMM; 加密视频识别

中图法分类号: TP393

中文引用格式: 吴桦, 罗浩, 赵士顺, 刘嵩涛, 程光, 胡晓艳. 面向HTTP/2流量多路复用特征的加密视频识别方法. 软件学报, 2025, 36(7): 3375–3404. http://www.jos.org.cn/1000-9825/7236.htm

英文引用格式: Wu H, Luo H, Zhao SS, Liu ST, Cheng G, Hu XY. Encrypted Video Identification Method for HTTP/2 Traffic Multiplexing Features. Ruan Jian Xue Bao/Journal of Software, 2025, 36(7): 3375–3404 (in Chinese). http://www.jos.org.cn/1000-9825/7236.htm

Encrypted Video Identification Method for HTTP/2 Traffic Multiplexing Features

WU Hua[1,2,3], LUO Hao[1], ZHAO Shi-Shun[1], LIU Song-Tao[1], CHENG Guang[1,2,3,4], HU Xiao-Yan[1,2,3]

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

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

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

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

Abstract: The rise of video platforms has led to the rapid dissemination of videos, integrating them into various aspects of social life. Videos transmitted in the network may include harmful content, highlighting an urgent need for cyberspace security supervision to

* 基金项目: 国家重点研发计划 (2021YFB3101403)
收稿时间: 2023-03-15; 修改时间: 2023-12-21, 2024-04-15; 采用时间: 2024-06-08; jos 在线出版时间: 2024-11-18
CNKI 网络首发时间: 2024-11-20

accurately identify harmful videos that are encrypted and transmitted in the network. The existing methods collect traffic data at main network access points to extract the features of encrypted video traffic and identify the harmful videos by matching the traffic features based on harmful video databases. However, with the progress of encryption protocol for video transmission, HTTP/2 using new multiplexing technologies has been widely applied, which makes the traditional traffic analysis method based on HTTP/1.1 features fail to identify encrypted videos using HTTP/2. Moreover, the current research mostly focuses on videos with a fixed resolution during playback. Few studies have considered the impact of resolution switching in video identification. To address the above problems, this study analyzes the factors that cause offsets in the length of the audio/video data during the HTTP/2 transmission process and proposes a method to precisely reconstruct corrected fingerprints for encrypted videos by calculating the size of the combined audio and video segments in the encrypted traffic. The study also proposes an encrypted video identification model based on the hidden Markov model and the Viterbi algorithm by using the corrected fingerprints of encrypted videos and a large plaintext fingerprint database for videos. The model applies dynamic planning to solve the problems caused by adaptive video resolution switching. The proposed model achieves identification accuracy of 98.41% and 97.91% respectively for encrypted videos with fixed and adaptive resolutions in 400000-level fingerprint databases, namely Facebook and Instagram. The study validates the generality and generalization of the proposed method using three video platforms: Triller, Twitter, and Mango TV. The higher application value of the proposed method has been validated through comparisons with similar work in terms of recognition effectiveness, generalization, and time overhead.

Key words: HTTP/2; DASH; large database; corrected fingerprinting; HMM; encrypted video identification

近年来, 随着移动互联网的发展, 视频应用已经成为互联网中的主流应用. YouTube、Facebook、TikTok 和抖音等国内外视频分享平台为用户提供了便捷的视频分享和转发功能. 根据 Ericsson Mobility 网站[1]的报告, 2023 年全球所有移动网站流量中约有 73% 是视频流量, 预计到 2028 年底, 这一比例将达到 80%, 其中来自社交媒体平台的视频流量占比最高. 互联网中传播的视频已经深度渗透到网民的社会生活中.

由于视频平台和社交平台中的视频来源多样, 如果平台审核不及时, 各平台提供的视频也会包含部分色情、暴力、谣言等类别的有害视频, 本文称为公害视频, 这些公害视频给网络空间和社会造成了严重的不良影响. 然而, 公害视频因其数量庞大、制作成本低、传播速度快、加密传输等特点[2], 给监管造成了极大的困难.

网络监管部门为了防止公害视频的传播, 可以采取多种方式进行监管. 对一些专门提供公害视频的平台, 当监管部门识别出这些视频平台的 IP 地址或域名后, 如果视频平台在本网络管理域范围内, 可以对视频平台进行处罚和治理; 如果视频平台不在本网络管辖域范围内, 可以根据 IP 地址或域名对其在本管理域内的传播进行阻断. 但是, 互联网中绝大部分视频平台是正常视频平台, 在大量的正常视频中只有少量的公害视频. 在这样的现实场景中, 如果能精准地识别视频内容, 就可以对公害视频的传播进行细粒度管控, 对正常的视频流量则无须管控, 从而达到精细化网络管理的目的. 为达成这一目标, 需要及时识别出被传输的公害视频.

目前, 全球主流视频平台均已采用加密技术来传输视频数据. 根据 W3Techs 网站[3]发布的报告, 在全球网站中, 默认使用 HTTPS 等加密协议的网站比例已从 2023 年 1 月的 81.5% 上升至 2024 年 1 月的 85.6%. 随着互联网中加密流量占比的提升, 尤其是加密视频流量占比的迅速提升, 加密传输技术给普通用户带来安全保护的同时, 也给不法分子套上了一层加密外衣[4,5], 导致网络监管机构对网络环境的监管难度成倍提升.

根据数据来源的不同, 现有对公害视频进行识别的方法主要有两类. 第 1 类方法是分析视频平台的视频文件, 通过深度学习对视频中的图像进行学习[6,7], 基于训练出的模型对未知视频抽取画面帧进行内容识别, 从而对识别出的公害视频进行传播阻断. 这类方法的数据源是视频平台的视频文件, 适用于视频平台的管理者进行内容审查, 但是这类方法需要的硬件资源价格昂贵, 很多小的平台迫于成本和技术的限制无力实施, 也有一些视频平台主观上不愿意进行内容审查, 导致网络公害视频泛滥. 第 2 类方法是分析在网络主要接入点采集的流量数据, 提取加密视频流量的特征[8,9], 基于已有的公害视频库, 通过流量特征的匹配识别出被传输的公害视频. 这类方法不需要寻求平台的合作, 监管部门部署时具有很好的可控性. 难点在于, 由于网络加密传输协议的持续演进, 已有的方法无法分析使用新协议传输的数据. 使用多路复用技术传输的协议, 如 HTTP/2, 已经被广泛部署并极大地改变了加密流量的传输特点, 因此第 2 类方法需要对使用多路复用技术而产生的新流量特征展开分析.

目前已有不少从网络流量中识别视频的研究成果. 根据识别目标的不同, 可分为加密视频平台分类[10,11]、加

密视频类别分类[12]、加密视频服务质量识别[13,14]、加密视频播放模式识别[15]、加密视频内容识别[16,17]等. 这些研究中, 只有加密视频内容识别才能满足对网络公害视频进行监管的目的.

然而, 已有的加密视频内容识别研究还存在一些普遍性的问题. 第一, 由于采集大型加密视频数据集工作量巨大, 费时费力, 所以当前研究大多使用自建的小型或微型数据集展开研究和验证, 也有少数研究是面向大型数据集进行的, 但使用的是模拟的大型指纹库, 这导致现有研究缺少面向大型真实数据集的验证. 第二, 目前针对加密视频内容的识别主要基于 HTTP/1.1 协议, 近几年来, 随着 HTTP/2、QUIC 和 HTTP/3 等多路复用协议的快速普及, 加密视频流量传输特征发生了巨大的变化, 传统基于 HTTP/1.1 协议的识别方法已无法适用于使用新型多路复用协议的场景. 第三, 当前对加密视频识别的研究大多针对的是播放时分辨率固定的视频, 很少考虑到流媒体自适应播放时分辨率切换给识别带来的影响.

针对以上问题, 本文基于自采集的大型真实数据集, 研究使用 HTTP 自适应流媒体 (HTTP adaptive streaming, HAS) 技术进行视频分发, 并使用多路复用的 HTTP/2 传输的加密视频识别问题.

本文主要贡献如下.

(1) 针对当前没有公开的大型加密视频数据集的问题, 同时为了在大型数据集中验证本文的方法, 设计实现了自动化的加密视频指纹采集系统, 构建了一个包含 40 万条视频明文指纹的大型数据集.

(2) 针对多路复用传输协议导致的加密视频传输数据难以精准还原问题, 本文深入研究了 HTTP/2 协议数据加密传输过程中, 数据长度发生偏移的原理, 提出从加密传输的组合音视频数据单元中精准修正还原出加密视频修正指纹的方法, 使用 Facebook 和 Instagram 视频平台测试集的实验结果表明, 超过 99.70% 的指纹还原误差都在 0.5‰范围内.

(3) 针对在大型指纹库中识别速度慢以及视频分辨率自适应切换导致视频识别难的问题, 本文提出以加密视频修正指纹和视频明文指纹库为基础, 以动态滑动窗口匹配机制、隐马尔可夫模型和维特比算法为实现方法, 构建加密视频识别模型, 最终在 40 万级的真实视频明文指纹库场景中实现了超过 98.23% 的识别准确率, 且能克服视频播放过程中分辨率自适应切换的影响, 与其他识别研究方法相比, 本方法具有明显的优势.

本文第 1 节介绍加密视频内容识别领域中的相关工作. 第 2 节介绍加密视频识别过程中的背景技术. 第 3 节介绍加密视频识别的原理和本文方法的主要架构. 第 4 节介绍本文构建大型数据集的方法. 第 5 节基于大型真实数据集, 给出面向 HTTP/2 多路复用传输的加密视频数据精准修正还原方法, 进而构建出加密视频修正指纹. 第 6 节提出利用加密视频修正指纹和视频明文指纹快速识别出加密视频的方法. 第 7 节给出本文方法在 40 万级的大型视频指纹库场景下的评估结果与理论分析. 第 8 节给出与类似工作的对比实验结果. 第 9 节对研究进行总结并展望未来的工作.

1 相关工作

从网络流量中识别加密视频内容是网络监管中的热点和难点问题. 针对此问题, 国内外都已有一些研究成果, 从方法上可分为两类. 第 1 类利用人工智能技术, 使用机器学习或深度学习, 直接从视频流信息中提取流量特征, 送入机器学习模型, 使用大量加密视频传输数据进行训练, 最终通过模型预测视频内容, 简称机器学习模型预测法. 第 2 类是将领域知识和人工智能相结合, 通过带外方法构建视频指纹库, 监测时从视频流中尽量复原出指纹特征, 与指纹库中的视频指纹进行匹配, 从而识别视频, 这类方法简称为指纹识别法.

第 1 类方法使用机器学习模型预测视频内容, 主要的识别思路是将视频播放时的加密音频或视频数据流作为输入, 使用各种机器学习算法训练音视频流的分类器, 训练好的分类器可以用来识别音视频. 选择合适的特征, 可以实现对加密流量的有效分类[18]. Dubin 等人[19]提出了一种使用分类方法识别加密 HTTP 自适应视频流标题的算法. 该算法的识别思路是利用加密视频的流量模式, 结合 SVM 机器学习算法, 在包含 100 个视频的 10 000 个 YouTube 传输实例的数据集中进行实验. 该方法的分类准确率为 95%, 并且可以将不在训练集中的视频分类为未知视频. 但是, 该算法需要对每个待识别视频进行 100 次的播放用于学习, 训练的代价过大. 此外, 如果在真实网络

中部署该方法, 当视频被用户在不同的环境中播放从而呈现出新的流量特征时, 由于缺少训练样本, 分类器有极大可能无法正确识别出视频内容. Liu 等人[20]提出了一种基于加密视频流间歇性特征的机器学习模型识别方法. 该方法利用自适应流媒体技术在传输视频时所呈现的间歇性流量模式特征作为输入, 以 KNN 算法作为分类器, 实现了良好的识别准确率和召回率. 然而, 为了获取稳定的传输特征, 该方法需要长时间重复采集加密视频传输数据. 此外, 相同视频在不同网络环境中的数据特征并不一样, 因此会导致模型的失效. 由此可见, 使用机器学习模型预测视频内容的方法需要大量的训练样本, 并且训练出的分类器无法应用到不同的数据传输环境中.

第 2 类方法将领域知识与机器学习结合, 利用带外构建的加密视频指纹库识别被传输的加密视频, 被称为基于视频指纹的识别方法. 根据加密视频指纹库构建方法的不同, 可以细分为明文指纹识别和密文指纹识别两种类型.

明文指纹识别方法是指通过带外的方式获取可以唯一标识视频内容的明文信息作为视频指纹, 构建视频明文指纹库. 对于需要识别的加密流量, 通过对加密视频传输数据进行修正还原构建视频修正指纹, 将视频修正指纹与明文指纹库中的明文指纹匹配, 就可以识别出该视频内容. 这类方法的关键技术难点是需要准确地从加密流量中还原出对应的明文指纹, 修正误差过大会导致当指纹库规模增大时的匹配失败和误判增加. Xu 等人[21]在其研究中针对使用动态比特率 (variable bitrate, VBR) 编码的视频, 提出了一个系统 CSI. 该系统通过分析加密视频数据推断出用户下载的音视频数据片段的大小, 即修正还原出音视频片段的大小, 从而作为视频修正指纹用于视频识别. 但该系统对于使用 HTTPS 和 QUIC 协议的加密视频数据的还原误差较高, 分别为 1% 和 5%, 对于一个常见的时长为 5 s、分辨率为 720p 的视频片段, 其大小约为 0.5 MB, 1% 的误差意味着真实值和还原值之间约有 5 000 B 的偏差, 由于片段通过长度匹配, 5 000 B 的误差将会导致视频片段非常容易被误识为其他视频片段, 在 Xu 等人的后续实验中, 当还原误差为 5% 时, 利用连续 6 段视频片段对视频进行标识的准确率仅有 90%, 并且没有给出假阳率指标, 这也说明了 CSI 系统的还原方法并不可行. Reed 等人[17]提出了一种能够识别 HTTPS 加密的 Netflix 视频的方法. 该方法将视频片段大小和顺序作为视频明文指纹, 并通过爬虫等方式构建了一个包含 33 万个视频明文指纹的指纹库. 该方法主要是使用 Adudump 工具从加密视频流中提取特征构建加密视频传输指纹, 虽然考虑到 HTTP 头部和 TLS 协议对数据长度造成的影响, 但并未给出具体的解决方法. 该研究在评估实验效果时只给出了识别准确率, 缺少精准率、召回率等指标数据, 也并未给出能够全面评估算法有效性的 $F1$ 得分. Wu 等人[22]提出了面向 HTTP/1.1 协议的精准指纹还原方法, 实现了在大型指纹库场景中进行加密视频识别的方法, 并验证了该方法的准确率、查准率、查全率、假阳率指标. 但是, 该方法使用的大型指纹库为模拟指纹库, 且该方法针对的是通过 HTTP/1.1 协议传输的加密视频流量, 与 HTTP/2 多路复用的流量特征有较大差别, 由于现网中使用 HTTP/2 协议传输的视频流量逐渐增多, 因此需要针对使用 HTTP/2 协议的视频平台展开研究.

当难以获得视频明文信息来构建视频明文指纹库时, 还可以通过构建视频密文指纹库实现加密视频的识别, 即密文指纹识别方法. 视频密文指纹识别方法是从采集到的加密视频传输数据里提取传输流量特征, 将提取到的传输特征和视频名称构成一个传输实例并作为视频密文指纹, 从而构建视频密文指纹库. 需要进行识别时, 与密文指纹具有相似传输特征的视频被判断为同一个视频. Gu 等人[23]提出了一种通过加密视频流的侧信道攻击方式识别视频的方法. 该方法按照固定时长将视频传输数据分段, 并将这些传输数据的段落流量特征作为视频指纹. 由于其指纹不针对视频明文片段进行长度还原, 仅提取密文特征, 因此属于密文指纹. 该方法利用在侧信道采集的加密视频数据传输特征构建视频密文指纹库, 识别视频时, 利用视频传输指纹与密文指纹库中的密文指纹进行比对, 从而识别视频. Afandi 等人[24]也采用了密文指纹识别视频的方法, 考虑到影响密文指纹稳定性的因素, 采用了一种差分指纹作为视频的密文指纹, 这种方法在一定程度上消除了由于视频分辨率自适应性切换和网络环境变化造成的加密视频流量特征前后的不一致性, 保证了在固定周期内的流量特征的相对稳定性. 在进行验证时, 他们采集了 86 组视频传输数据, 其中一半是 VPN 流量, 另一半是正常的 HTTPS 流量, 测试结果显示: 对于非 VPN 流量, 其分类准确率为 90%; 对于 VPN 流量, 其准确率仅有 59%. 该研究的实验数据集偏小, 且识别的准确率不高. Bae 等人[25]提出了一种针对 LTE 网络的视频识别攻击方法. 该方法以提取的加密视频流量特征作为视频密文指纹, 在包含 100 个视频的加密视频数据集中可以实现 97.2% 左右的识别准确率. 但是, 为了提取稳定的传输特征构建视频密文指纹库, 该方法需要在相同 LTE 网络中重复采集 30 余次目标视频传输流量. 在使用密文指纹识别加密视频的

方法中, 由于基于 HTTP 协议的自适应流媒体协议能够根据网络环境的变化自适应地选择不同质量的音视频片段下载, 导致每次传输的数据都可能不同, 密文指纹的稳定性难以保证, 因此如何构建视频密文指纹库是一个难点.

基于上述分析, 由于基于视频指纹的识别方法中的明文指纹识别法所使用的明文指纹不受网络环境的干扰, 只要准确地还原构建出视频修正指纹, 就有可能达到理想的识别效果, 因此本文选择使用明文指纹识别法开展加密视频内容的识别研究. 虽然已有使用明文指纹识别法对加密视频内容进行识别的研究, 但它们仍然存在一些普遍性的问题: 第一, 很少有使用大规模真实数据集进行研究和验证的结果, 大部分研究数据集较小, 少量面向大规模数据集的研究是基于模拟的大规模指纹库; 第二, 互联网上传输视频的 HTTP 协议已经逐步从 HTTP/1.1 发展为 HTTP/1.1、HTTP/2、QUIC 等多种协议并存, 新型协议使用的多路复用技术进一步混淆了加密视频的流量特征, 已有加密视频识别技术都利用了 HTTP/1.1 传输模式特征, 这导致这些技术不能用于识别使用多路复用传输技术的加密视频; 第三, 已有研究都没有考虑自适应流媒体分辨率切换技术给视频识别带来的影响, 这也影响了方法的实用性. 针对当前研究中存在的问题, 本文展开面向 HTTP/2 多路复用特征的分辨率自适应切换的加密视频内容识别研究, 并使用 40 万级的真实视频指纹库进行验证.

2 背景技术

2.1 HTTP 自适应流媒体技术

基于流量分析的加密视频识别方法与视频所使用的流媒体技术具有重要关系. 为了给具有动态网络条件和异构设备的用户提供稳定高质量的视频流服务, 当前全球主流的视频平台都使用了 HTTP 自适应流媒体技术[26], 如 MPEG 牵头开发的基于 HTTP 的自适应码率流媒体技术 MPEG-DASH[27]以及苹果公司制定的 HLS (HTTP live streaming) 技术[28]. 这些技术基于 HTTP 协议, 对用户防火墙和网络地址转换 (NAT) 技术十分友好, 适合内容分发网络 (CDN) 以缓存方式分发. HAS 技术中使用最广泛的是 MPEG-DASH (简称为 DASH) 技术. 下面以 DASH 为例介绍 HTTP 自适应流媒体技术.

DASH 2011 年成为国际标准 ISO/IEC 23009-1, 并于 2012 年 4 月正式发布. 目前, DASH 已经成为主流的视频传输标准, 被大量视频内容提供商所采用[29].

在一个典型的 DASH 视频传输系统中 (如图 1 所示), 在服务端, 通过 VBR 算法将视频内容提前编码为不同比特率的多种质量版本, 每个编码后的视频被切割为若干段时长相等的视频片段[30], 一般为 2–10 s, 且不同比特率视频的各个片段在时序上对齐, 以便在网络条件发生变化时, 快速地在不同比特率的片段之间自由切换. 此外, 在使用 VBR 编码视频时, 还会生成与之对应的媒体描述 (media presentation description, MPD) 文件[31]. 该文件是一种 XML 格式的清单文件, 视频的各种内容信息, 如视频简介、元数据、编解码器、字节范围、媒体类型和音视频资源地址等都被记录在 MPD 文件中.

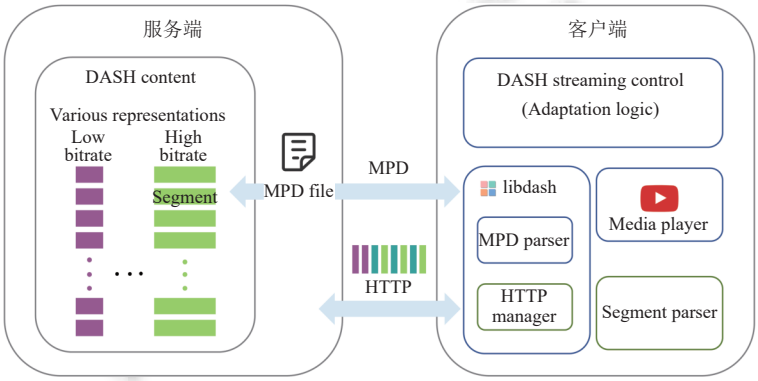

图 1 DASH 视频传输系统

当打开视频时, 客户端首先请求获取 MPD 文件, 并对其进行解析, 进而获取该视频的各种比特率资源的地址和描述信息, 然后根据自身的网络状况, 选取合适的视频片段进行下载. 在视频播放的同时, DASH 客户端可使用 ABR 算法, 根据网络状态、应用状态和客户端缓冲区状态动态地切换不同比特率的视频片段, 从而实现自适应码率播放.

虽然自适应流媒体技术提高了视频用户的体验, 但其将视频片段顺序传输的特点也为研究者从流量中通过侧信道技术分析出视频内容提供了可能.

2.2 HTTP/2 协议

由于 HTTP 自适应流媒体技术均基于 HTTP 协议构建, HTTP 协议作为视频数据传输的载体, 其协议规范和传输机制直接影响着加密视频流的传输特征, 与加密视频识别技术密切相关. HTTP 协议最初是专门被设计用于在网站服务器和客户浏览器之间传输 Web 应用数据的协议, 由于其应用简单方便, 易于部署, 且 HTTP 流量通常不会被各类防火墙过滤, HTTP 协议已经不局限于传输 Web 应用数据, 而是广泛用于传输自适应流媒体、各类 APP 数据等. 随着基于 HTTP 协议的应用越来越多, 为了提高网络传输性能, 优化用户服务体验, HTTP 协议一直不断进行改进, 目前主要被使用的 HTTP 版本包括 HTTP/1.1, HTTP/2, QUIC 和 HTTP/3.

HTTP/1.1 协议自 1999 年发布以来, 在很长一段时间内都是使用最广泛的 HTTP 协议版本. 然而, 随着互联网的飞速发展, HTTP/1.1 因串行通信、队头阻塞和头部冗余等性能问题逐渐不能满足现代 Web 应用对高并发和大数据量传输的需求. 为了解决这些问题, 互联网工程任务组 IETF (Internet engineering task force) 在 2015 年 5 月发布了 HTTP/2 协议. 作为 HTTP/1.1 协议的全新升级版本, 它从根本上解决了 HTTP/1.1 的性能瓶颈, 提供了更加快速、安全、高效的网络通信能力, 并迅速获得了全球众多浏览器和服务器的支持. 截至 2023 年 12 月, HTTP/2 在全球互联网网站中的使用率已达到了 37.6%[32], 在 TOP 10 000 的网站中使用率更是高达 62.3%. 几乎在同一时期, Google 于 2015 年提出了基于 UDP 的 QUIC 协议, 旨在克服 TCP 协议的队头阻塞、连接延迟和连接迁移等固有问题, 以实现更快的连接建立和无缝的连接迁移. 基于 QUIC 的 HTTP/3 协议, 最初被称为 HTTP over QUIC, 于 2022 年 6 月完成标准化, 旨在进一步提升网络性能和用户体验, 目前已经初步在全球范围内推广应用.

当前, 互联网中呈现出 HTTP/1.1、HTTP/2、QUIC 和 HTTP/3 协议并存的现象. HTTP/2 正逐步取代老旧的 HTTP/1.1, 成为应用最广泛的 HTTP 协议版本. 与此同时, 以谷歌为代表的服务提供商已经开始在其服务中部署 HTTP/3 协议. 然而, 有研究指出[33], 在网络服务质量不佳的情况下, HTTP/3 的传输效率不如 HTTP/2. 因此, 服务提供商选择同时支持多种协议, 并根据用户的地理位置和网络环境来选择使用 HTTP/2 或 HTTP/3. 例如, Facebook 在网络状况良好时优先使用 HTTP/3 进行视频传输, 而在网络延迟较大的路径上则切换至 HTTP/2 协议. 因此, 展望未来, 可以预见 HTTP/1.1 的使用将逐渐减少, 而 HTTP/2 和 HTTP/3 将在很长一段时间内在互联网中并存使用. 本文研究使用 HTTP/2 协议传输的加密视频内容的识别问题.

HTTP/2 并非 HTTP/1.1 的简单升级版, 而是使用了多路复用技术, 替代了 HTTP/1.1 的传输模式. HTTP/2 引入了二进制分帧、多路复用、首部压缩、服务器推送和流的优先级控制等功能[34], 从根本上解决了 HTTP/1.1 的诸多问题, 大幅提升了传输数据的性能. 其中, 二进制分帧是 HTTP/2 所有性能提升的核心. 它在不改变原有 HTTP 语义的基础上重新定义了数据的封装和传输, 并将帧作为最小的数据单元, 所有的帧都采用二进制编码, HTTP/2 消息由各种数据帧共同构成. 如图 2 所示, HTTP/2 协议将 HTTP 首部信息通过 HPACK 技术压缩后封装为一个 HEADERS 帧进行单独传输, 并将 HTTP 负载数据封装为若干个 DATA 帧, 此外还有 WINDOW_UPDATE 帧等控制帧. 在 HTTP/2 中, 每个请求或响应使用不同的流 (默认至多 100 条流). 通过使用二进制分帧, 给每个帧分配一个流标识符, 使各个帧可以交错传输, 以支持同时发送多个独立请求, 当接收到流的所有帧时, 接收方可以将帧重新组合成完整的消息. 这使得 HTTP/2 可以实现多路复用, 在使用单条 TCP 连接且不发生阻塞的情况下, 可以将多个请求和响应并行处理, 以此大幅减少网络延迟, 提高网络资源的利用率.

HTTP/2 协议虽然在正式规范 RFC 7540[35]中并没有强制使用 HTTPS 加密传输, 但是在实际的使用过程中,

HTTP/2 协议都是基于 HTTPS 实现的, 这也成为事实上的标准. 研究人员无法根据有效载荷来研究 HTTP/2 的流量, 只能通过分析加密流量的传输特征得到信息.

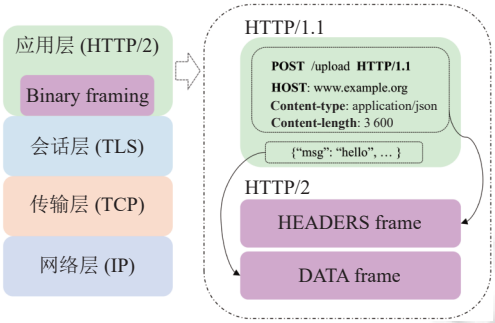

图 2 HTTP/2 二进制分帧层

多路复用技术在给 HTTP/2 数据传输带来极大的效率提升的同时, 也给网络流量研究人员带来了新的挑战. 多路复用技术使得 HTTP 流量特征发生很大的变化, 之前基于 HTTP/1.1 传输模式建立的流量模型将不再适用于 HTTP/2 的流量, 因此已有面向 HTTP/1.1 协议的视频识别方法已无法适用于使用 HTTP/2 传输的视频. HTTP/2 中新的协议功能给加密视频识别带来了新的挑战.

3 加密视频识别方法整体架构与关键技术难点

本节给出本文方法的整体架构, 并明确方法中需要解决的关键技术难点.

根据第 1 节的分析, 只有使用明文指纹识别方法才有可能实现大规模加密视频场景中的精准加密视频识别. 因此本文方法使用明文指纹识别法. 为方便后续叙述, 本文明确如下定义.

定义 1. 明文指纹: 通过使用稳定性较强, 且未加密的视频明文信息构建起来的视频指纹.

定义 2. 明文指纹库: 明文指纹的集合.

定义 3. 传输指纹: 从侧信道提取的加密视频流的流量特征序列, 也称密文指纹.

定义 4. 密文指纹库: 传输指纹 (密文指纹) 的集合.

定义 5. 修正指纹: 通过特定修正还原算法, 还原加密视频数据长度, 将视频传输指纹修正还原形成的指纹, 修正还原算法的还原精度越高, 修正指纹质量就越高, 视频修正指纹和视频明文指纹也就越接近.

在加密视频传输场景中, 由于无法直接利用视频应用层信息, 所以对加密视频的识别主要利用音视频数据应用层数据单元的长度和传输顺序. 如第 2.1 节所介绍, 当前全球主流的视频平台都使用了 HAS 技术. 以 DASH 为例, 为了实现在视频播放过程中视频分辨率的自适应切换, DASH 使用可变比特率编码技术, 将音视频文件进行编码, 形成不同分辨率质量的视频, 同时将编码后视频切割为时长相等的视频片段. 由于视频内容画面的复杂度和视频片段大小具有强相关性[36,37], 因此视频内容不同, 视频片段的大小也会呈现出较大的差异. 用户请求播放视频时会根据网络条件依次请求下载各种质量的视频片段. 由于这种视频片段的长度和传输顺序与视频内容具有强关联性, 且视频在上传时经过一次编码后一般不会再次编码, 具有很强的稳定性, 因此可以将视频的这种片段的原始长度序列作为标识视频内容的明文指纹. 由于视频内容不同会导致每个视频的片段长度序列不同, 因此如果能从传输数据中准确地获得这些片段序列的长度, 就能在不解密的情况下识别加密视频.

视频的明文指纹指视频片段的原始长度序列, 是视频资源在应用层的特征. 在网络中传输时, 这些音视频片段会被各种协议层层处理, 附加上各种传输和安全管理信息, 导致实际传输的数据量要比原始明文数据量略大, 构成视频传输指纹. 在明文指纹库识别场景中, 需要将视频传输指纹通过设计的修正还原方法处理, 去除掉各种附加管理信息, 形成视频修正指纹, 最后利用合适的指纹匹配算法, 将还原出的修正指纹和明文指纹库里的视频指纹进行匹配, 确定视频内容标签.

本文使用明文指纹识别加密视频的方法如图 3 所示. 首先利用自动化采集系统, 在真实播放场景中采集视频明文指纹和加密视频传输数据, 构建视频明文指纹库, 其次从加密视频数据流中提取数据特征, 经修正还原模型处理后构建出加密视频修正指纹, 最后以加密视频修正指纹和视频明文指纹库为基础, 设计合适的视频识别方法, 进而识别加密视频内容.

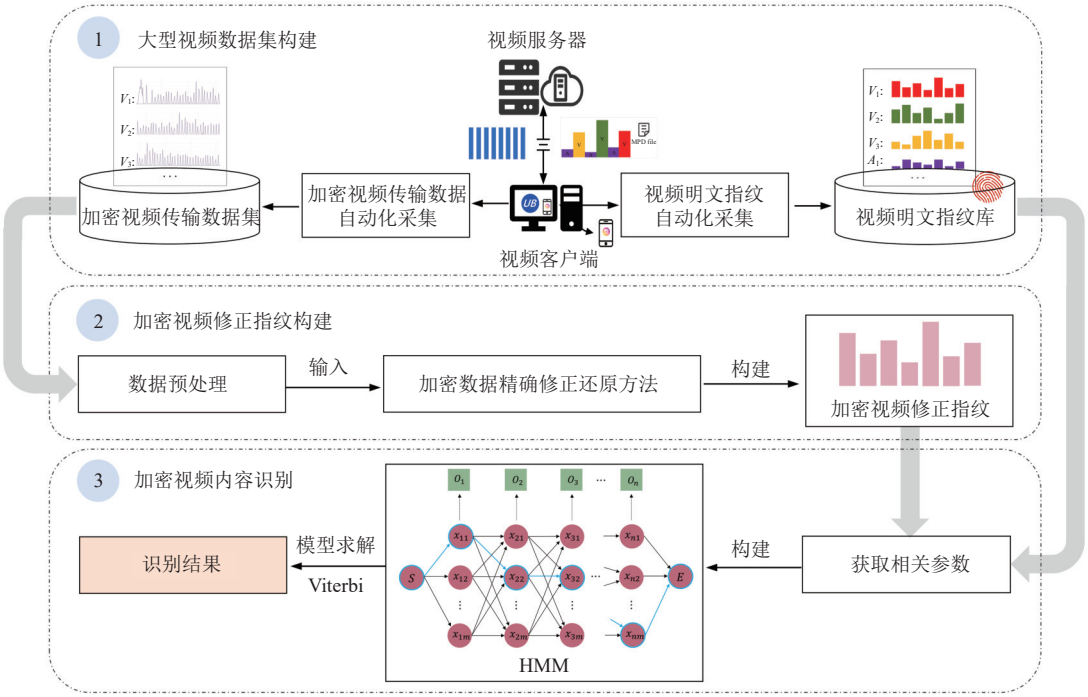

图 3 使用明文指纹识别加密视频方法的整体架构

在图 3 所示的识别过程中, 第 1 步的明文指纹库构建过程中得到的明文指纹是准确的. 但是第 2 步从密文中还原出修正指纹, 需要考虑到加密和传输音视频数据时的多种影响因素, 不同的影响因素以及不同的修正还原技术会得到不同的还原结果, 这就导致修正的指纹结果是不确定的. 这种不确定性也会进一步影响第 3 步视频识别的准确性. 因此需要分析各层网络协议, 特别是 HTTP/2 协议对修正还原指纹的影响.

本文的方法需要解决的关键技术难点主要有以下 3 点.

(1) 各层协议对音视频片段的封装和切分导致加密数据长度相对于原始明文长度的随机增加.

在传输过程中, 音视频数据依次经过应用层 HTTP 协议、TLS 协议、TCP 协议和 IP 协议的层层处理, 每层协议都会给原有的音视频片段数据添加各自的协议头部, 并对音视频数据进行加密、压缩或填充处理, 因此在传输过程中, 音视频片段数据的传输长度与原有长度相比会增加, 并且增加的长度与每次视频传输时的设备状态、网络环境相关, 具有一定的随机性. 本文将在第 5 节分析各层协议对应用层数据单位的封装过程, 并给出对应的解决方案.

(2) HTTP/2 协议对流特征的改变导致需要新的视频修正指纹构建方法.

HTTP/2 协议引入了首部压缩、服务器推送和多路复用等新功能以提高传输效率, 但是其中的首部压缩和多路复用功能使得之前针对 HTTP/1.1 的视频修正指纹构建方法无法用于 HTTP/2 视频流量. 在 HTTP/1.1 中, 首部数据的长度通常只分布在特定范围内, 而在 HTTP/2 中由于采用了首部压缩技术, 首部数据的长度分布出现了很大的变化, 所以在流量分析方法中, 过滤首部数据时所采取的方式也会有所变化; 除了首部压缩之外, 多路复用也是影响视频识别的主要因素, 直接导致了现有的针对 HTTP/1.1 的视频流分析方法无法适用于 HTTP/2 的视频流

量. 在 HTTP/1.1 中, 音视频数据往往会通过两条 TCP 流分别进行不同类型的数据传输, 以提高效率. 而在 HTTP/2 中, 由于多路复用功能的引入, 往往只需要建立 1 条 TCP 流就能够承载多种数据的交互传输, 由于音频数据和视频数据在同一条 TCP 流上交替传输, 因而两次请求之间的响应数据往往是同时包含音频和视频的混合数据, 且由于流量是加密的, 这给数据长度的修正带来较大的困难, 本文将在第 5 节中对此展开研究.

(3) 自适应分辨率切换和用户拖动播放导致视频片段不按序传输.

在视频播放过程中, 可能会发生分辨率自适应切换事件和用户手动改变播放进度事件, 这使得重复播放同一视频时的传输数据特征会发生较大变化, 从而严重影响依靠侧信道特征识别加密视频的准确率. 当固定分辨率播放时, 多次反复播放同一个视频, 服务器发送的是同一个分辨率的视频资源, 此时, 视频传输数据的流量特征相对稳定, 只会因为网络延迟和丢包等网络状况变化而发生变化, 可以使用协议相关的领域知识去除这些干扰. 而采用自适应分辨率播放时, 网络状况的变化会触发分辨率的自适应切换, 实际中多次播放同一个视频时传输的可能是不同分辨率的视频数据, 这就导致传输过程中的数据量、传输模式会发生很大变化, 从指纹层面上看, 同一视频不同分辨率的指纹是不同的, 切换分辨率也就改变了对应视频的明文指纹, 同样, 用户手动改变播放进度也会使得视频片段不按顺序下载, 导致与之对应的明文指纹也发生变化, 因此, 在实际视频播放过程中, 修正指纹和明文指纹很难完全匹配, 只有选用鲁棒性强的指纹匹配识别算法, 针对上述播放事件进行优化才能实现良好的识别效果, 本文将在第 6 节给出加密视频指纹匹配识别算法.

4 大型视频数据集自动化构建方法

为了弥补现有视频识别领域缺少真实大型视频数据集的不足, 并使得本文方法能够在真实场景中得到验证, 本文设计实现了一个大型视频数据集自动化构建系统, 如图 4 所示, 该系统包括加密视频明文和加密视频传输数据的全自动采集, 通过该系统可以自动构建出大型的真实视频数据集.

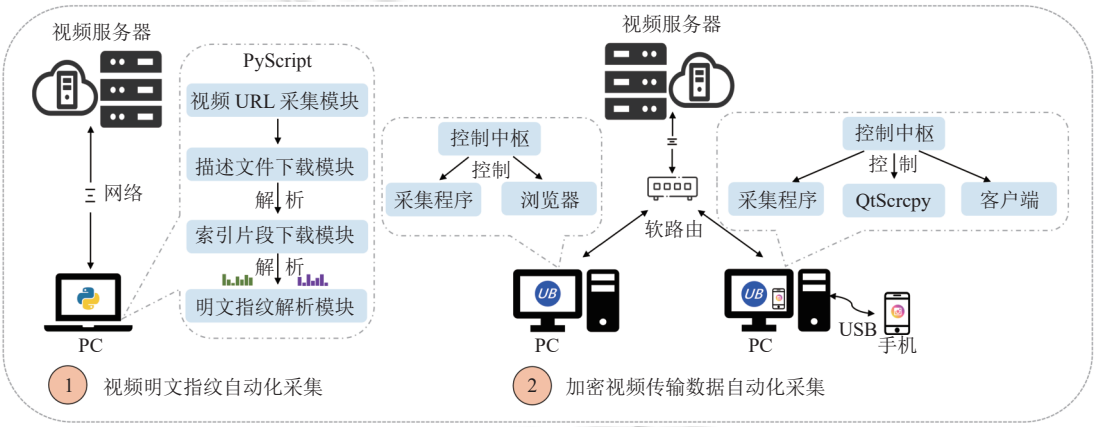

图 4 大型视频数据集自动化构建系统

(1) 视频明文指纹采集

视频明文指纹是指视频明文数据片段长度序列. 视频的明文指纹信息来自对视频的描述文件. 在使用 DASH 流媒体协议的播放场景中, 视频客户端会先向视频服务器请求当前播放视频的视频描述文件 MPD. 该文件中记录了对视频的各种描述信息, 其中可以用于构建明文指纹的信息是视频索引片段内的字节范围 (IndexRange) 信息. 视频索引片段[38]是 DASH 协议中定义的一种特殊的片段, 其中保存着视频经 VBR 编码算法切分后各视频数据片段的大小. 在视频播放过程中, 客户端会首先请求下载索引片段, 通过解析视频索引片段, 可以直接获取各视频片段的大小和字节范围, 结合视频资源地址, 即可实现独立请求下载各视频片段. 因此, 通过只下载并解析视频索引片段, 就能获取到各视频片段的长度信息, 从而得到视频的明文指纹.

如图 4 所示, 视频明文指纹自动采集系统主要包括视频 URL 采集模块、MPD File 下载模块、视频索引片段下载模块和明文指纹解析模块. 首先通过 URL 采集模块, 采集目标平台不同类别、时长和分辨率的视频 URL, 去重后作为输入送入 MPD File 下载模块, 通过视频 URL 和相关签名信息下载视频 MPD 文件, 解析 MPD 文件后提取关键信息, 最后根据提取的信息对索引片段解封装后提取视频明文指纹, 并将该过程中提取到的视频的关键信息和视频明文指纹作为一个整体存入视频明文指纹库中.

由于明文视频指纹的采集不需要下载和播放视频, 而是利用了视频片段的索引信息, 这些索引信息的数据量很小, 利用上述介绍的采集程序, 整个采集过程可以自动化实现, 无论从时间还是网络带宽耗费上都只需要很低的成本. 在实际采集过程中, 只需 1–2 s 就能采集一个视频的所有分辨率的明文指纹, 构建一个包含真实平台 40 万级的视频明文指纹库仅需 1–2 天. 此外, 本文使用的明文指纹库只需要存储视频的分段信息及视频标题等描述信息, 并不需要存储视频文件和视频播放的传输信息, 所需的硬盘存储空间也很小.

(2) 加密视频传输数据采集

加密视频传输数据集是在真实网络播放环境中采集的加密传输数据, 在本研究中作为训练集和测试集使用. 为了全面验证本文识别方法的有效性, 系统采集个人电脑 (PC) 的 Web 客户端和手机移动客户端两种终端播放场景下的视频加密传输数据.

对 PC 的 Web 客户端, 首先以机器人流程自动化 (robotic process automation, RPA) 技术平台 UiBot 作为自动采集系统的控制中枢, 模拟人工操作, 控制相关软件程序进行采集工作. 通过自动化程序, 以目标视频资源地址和相关信息作为输入, 同时控制流量采集程序和浏览器, 实现浏览器自动打开视频界面播放视频, 并控制视频客户端进行视频分辨率切换等操作, 与此同时, 控制流量采集软件采集当前所播视频的加密传输数据, 并将流量传输数据、传输数据描述信息共同作为视频传输实例存入数据库中.

手机移动客户端与 PC 的 Web 客户端类似, 同样以 PC 上运行的 UiBot 作为采集系统控制中枢, 控制相关软件客户端进行采集工作. 不同的是, UiBot 并不能直接运行在手机上控制手机进行相关操作, 需使用开源 Android 投屏软件 QtScrcpy 通过 USB 数据线将手机投屏至 PC 上, 配合 UiBot 实现模拟人工操作手机, 使用 Android 调试桥 (ADB) 以命令行的方式与手机系统进行底层交互, 控制手机进行相关网卡上的流量采集操作.

使用上述自动化采集系统, 可以实现不同视频终端的加密视频传输数据的全天候不间断自动采集, 本文成功构建了一个包含 40 万条视频明文指纹和加密视频传输数据的大型数据集. 该数据集的相关信息如表 1 所示.

表 1　大型视频数据集的相关信息统计

社交媒体	视频时长 (s)	视频类别	视频质量	加密视频传输数据 (个)	视频明文指纹 (万条)
Facebook	30–1 800	体育、音乐、综艺、动漫、动作、风景	144p、360p、720p、1 080p、1 440p、2 560p	4 630	31.035
Instagram				3 216	9.854

5　面向 HTTP/2 协议的加密视频修正指纹构建方法

视频明文指纹识别方法分为两个关键步骤. 第 1 个步骤是从加密视频传输数据中构建出精准还原的加密视频修正指纹; 第 2 个步骤是使用加密视频修正指纹与明文指纹库中的视频指纹进行匹配, 从而识别出视频. 本节面向 HTTP/2 协议传输的加密视频流解决第 1 个步骤中的技术难点.

原始的音视频数据片段在经真实网络环境中的各层协议处理之后, 其传输的数据长度会发生变化, 当使用原始音视频片段长度序列作为明文指纹识别视频时, 就需要对在实际网络中传输的加密音视频数据进行精准修正还原, 使其尽可能地接近原始数据长度. 只有精准还原的视频修正指纹才有可能从大型视频明文指纹库中准确地识别出视频. 本节分析干扰加密视频数据长度还原的因素, 针对各种干扰因素提出解决方法, 从而实现加密视频数据的精准还原, 构建出加密视频修正指纹.

本节提出的加密视频修正指纹构建方法的整体框架如图 5 所示. 该方法主要分为两个阶段. 第 1 阶段是模型

训练阶段, 该阶段利用加密传输数据和明文指纹库中对应的标签, 根据 TLS 和 HTTP/2 对数据进行加密封装和传输的原理, 依次训练 TLS 修正模型和 HTTP/2 修正模型, 用于修正加密传输数据. 第 2 阶段是利用第 1 阶段训练好的两个修正模型依次对加密视频传输数据进行精确修正还原, 从而构建出加密视频的修正指纹.

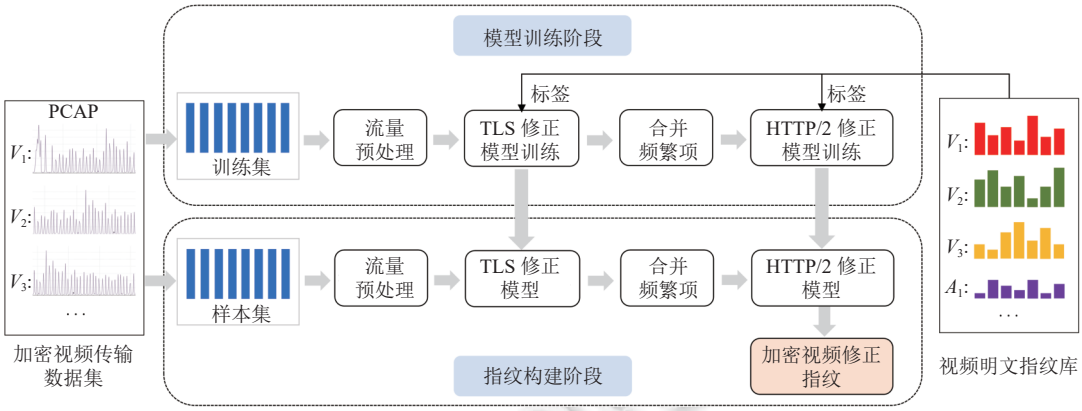

图 5 加密视频修正指纹构建框架

5.1 加密视频修正指纹构建基本思路

在使用明文指纹识别加密视频内容的过程中, 对加密传输数据还原的准确性直接影响到后续的识别效果, 而干扰还原准确性的因素主要是数据传输时所采用的各种协议封装. 本节将根据音视频数据在实际网络环境中的传输过程, 分析各种协议对所传输数据的影响, 从而确定干扰加密数据长度还原的主要因素, 并提出加密视频修正指纹构建的基本思路.

如第 2.1 节所介绍, 基于 HTTP 协议的自适应流媒体技术为实现视频播放过程中的不同视频分辨率的自适应切换, 使用 VBR 算法将视频编码为不同比特率的视频, 并将视频切分为时长相等的若干视频片段, 在实际视频播放过程中, 视频客户端会依据网络环境依次请求不同质量的视频片段和音频片段. 通过在 Chrome 浏览器开发者界面和代理软件中观察, Facebook 和 Instagram 在实际播放视频时, 客户端会连续请求多个视频和音频片段 (如图 6 所示), 组合方式包括多个视频、多个音频, 以及多个音频和视频的组合, 这些请求会得到连续传输的响应数据, 为方便后续介绍, 本文将这些连续传输的组合响应数据统称为组合数据单元 CDU (chunk data unit). 由于使用了多路复用技术, 这些 CDU 中的音频和视频分组加密后被混杂传输, 无法区分, 因此本文将 CDU 作为一个整体进行长度的修正还原.

加密视频修正指纹构建的基本思路是减去 CDU 在封装过程中增加的控制信息长度. 图 6 为音视频数据 CDU 被封装的过程. 在进入网络链路传输时, 音视频数据 CDU 会依次被 HTTP/2、TLS 和 TCP 等协议处理, 进行相应的切分、压缩、加密和填充等操作, 并添加对应的协议头部和控制信息, 以确保 CDU 的稳定可靠传输. 为了达到加密数据修正还原的目的, 需要从实际传输的负载长度中减去所有被添加的额外控制信息的长度, 可以得到音视频数据 CDU 的修正还原公式为:

$$\widetilde{CDU}_{\text{len}} = TCP\_Payload_{\text{len}} - TLS\_\theta_{\text{len}} \times N_{\text{TLS}} - H2\_HEADERS\_Frame_{\text{len}} - H2\_Frame\_Headers_{\text{len}} \times N_{H2} \quad (1)$$

其中, $\widetilde{CDU}_{\text{len}}$ 代表还原后的音视频 CDU 长度; $TCP\_Payload_{\text{len}}$ 代表传输该 CDU 数据的所有 TCP 报文段的有效负载之和; TLS_θ_{len} 代表单个 TLS 记录在承载传输数据后引入的干扰量的均值, 即单个 TLS 记录对所承载的传输数据的长度干扰值. 由于 TLS 协议具有数据压缩、填充和加密等机制 (压缩机制在 TLS 1.3 中已经被去除), 且会添加 TLS 头部等额外信息, 在未知加密套件及秘钥的情况下无法对负载数据进行还原, 只能通过线性回归的方式来整体评估 TLS 协议在负载数据前后引入的干扰量; N_{TLS} 表示 TLS 记录的个数; $H2\_HEADERS\_Frame_{\text{len}}$ 是指 HTTP/2 协议中 HEADERS 帧的长度, HTTP/2 协议有首部压缩的机制, 其使用 HPACK 算法对 HTTP 头部数据进

行压缩, 并作为 HEADERS 帧单独传输; $H2\_Frame\_Headers_{\text{len}}$ 代表一个 HTTP/2 帧的头部的长度; N_{H2} 指承载 CDU 数据的 HTTP/2 帧数.

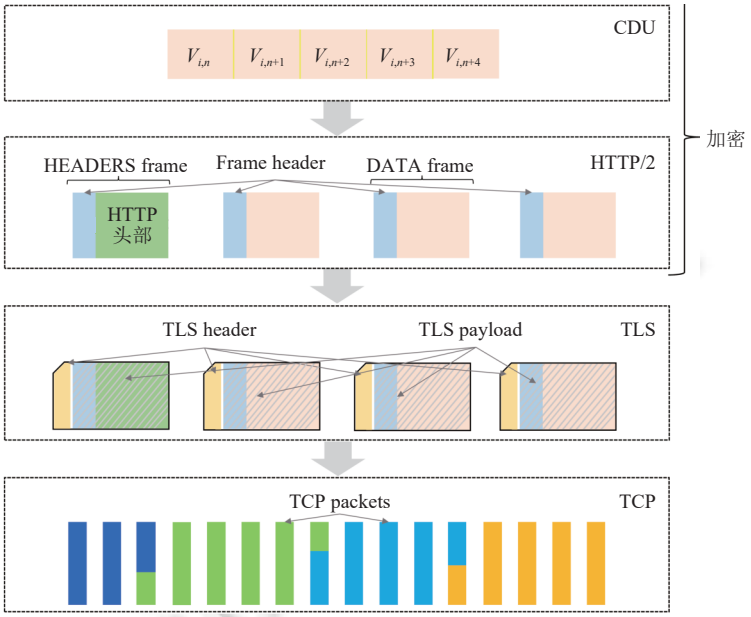

图 6 视频数据分组 CDU 封装过程

在公式 (1) 的诸多参数中, 一些参数可以通过直接或间接的方式从加密视频传输数据流中分析获得:

(1) $TCP\_Payload_{\text{len}}$: 由于 TCP 首部不是加密数据, 所以可以通过对承载音视频数据 CDU 的 TCP 分组的首部信息进行统计, 从而计算出总的 TCP 载荷长度.

(2) N_{TLS}: 虽然无法直接从 TCP 头部获取 TLS 记录数量, 但可以通过间接的方式计算出 TLS 记录个数. 由于 TLS 的“PlainText Header”结构中记录着加密数据块的长度信息[39], 并且这个结构的信息也是不加密的, 所以可以通过 TCP 负载数据中找到 TLS 记录的“PlainText Header”字段进行解析, 结合 TCP 分组的有效负载长度和序列号便可以得到 TLS 记录的个数.

(3) $H2\_HEADERS\_Frame_{\text{len}}$: 由于 HTTP/2 协议会使用 HPACK 算法将 HTTP 头部进行压缩, 封装成 HEADERS 帧进行单独传输, 且该帧的大小一般不会大于 1 个 TLS 记录的最大报文长度 MSS (maximum segment size), 所以该帧会单独使用一个 TLS 记录传输, 因此可根据时间戳、阈值和特征值来获取该参数.

(4) $H2\_Frame\_Headers_{\text{len}}$: 根据 HTTP/2 的协议标准 RFC 7540[35], 一个数据帧的头部长度为 9 B.

在获取到上述参数信息后, 修正还原公式 (1) 中只剩下参数 TLS_θ_{len} 和参数 N_{H2} 尚未获知, 要想计算这两个参数, 需要结合 TLS 协议和 HTTP/2 协议的特点, 从加密数据流中提取特征来构建相关修正模型, 此外这些特征与视频平台使用的 TLS 协议版本和参数设置有关, 因此需要根据不同视频平台的数据分别构建相关修正模型.

本文第 5.2 节给出从数据流中提取音视频数据, 划分出 CDU 的方法, 经过这个处理, 就可以提取出公式 (1) 中的 $TCP\_Payload_{\text{len}}$、N_{TLS}、$H2\_HEADERS\_Frame_{\text{len}}$ 和 $H2\_Frame\_Headers_{\text{len}}$ 这 4 个参数. 第 5.3 节针对 TLS 协议的干扰进行修正, 构建修正模型计算参数 TLS_θ_{len}. 第 5.4 节对 HTTP/2 协议的干扰进行修正, 构建机器学习模型计算参数 N_{H2}.

5.2 从加密视频数据流中提取组合数据单元

在加密视频数据传输过程中, 由于视频网页会加载页面、图标、CSS 和视频评论等信息, 导致采集到的流量数据中夹杂着大量的非音视频数据, 需要从这些混乱的数据流中区分出音视频流, 此外, 由于 HTTP/2 使用多路复

用技术, 在一个 CDU 中混杂传输的音视频无法分开, 本文的目标是将 CDU 作为一个整体进行长度修正, 因此, 需要在音视频流中精准划分出可以修正的 CDU.

(1) 音视频数据流提取

由于标准的 TLS 加密数据使用 TCP 的 443 端口, 所以首先可以从所有数据流中筛选出使用 443 端口的加密 TCP 数据流. 其次, 虽然 TLS 协议因其加密性使其负载数据内容无法获知, 但在建立 TLS 连接时, 其握手信息“Client Hello”中包含了一些未被加密的信息, 如服务器的 SNI (server name indication) 等, 可以利用 SNI 字段来区分音视频流和非音视频流. 需要说明的是, 由于 HTTP/2 有多路复用机制, 多个 HTTP 请求可以通过 1 条 TCP 流传输, 这意味着, 如果音频流和视频流的 SNI 相同, 那么音频数据和视频数据可以通过 1 条 TCP 流传输. 为了方便后续叙述, 本文将音频流、视频流、音频和视频流统称为音视频流. 最后, 可以根据流的数据量大小区分是否为视频的数据流. 经过以上过滤操作, 可以准确地提取出音视频流.

(2) TLS 记录序列提取

由于提取出的音视频流是位于传输层的 TCP 数据流, 因此还需要去除 TCP 数据包的头部信息长度, 并组合出 TLS 记录序列, 便于计算后续 TLS 记录在传输数据中产生的干扰量. 为此, 本文实现了一个类似于 TCP 协议栈端系统的缓冲机制, 将 TCP 报文根据其头部信息按序在缓冲区排序, 并基于 TLS 协议规范从 TCP 数据包中组合出 TLS 记录. 根据 TLS 记录头部仅存的明文信息, 可以得到每个 TLS 记录的长度信息, 再根据每个 TCP 载荷的实际长度信息, 将 TCP 数据包合并或者拆分到各 TLS 记录中, 构成 TLS 记录序列用于后续特征提取. 该过程可以有效确保 TLS 序列的完整性和准确性, 从而对抗真实世界中的网络抖动和拥塞现象.

(3) 精准划分加密音视频组合数据单元 CDU

音视频组合数据单元 CDU 可能包含 1 到多个音视频数据片段组合. 由于 HTTP/2 的多路复用特性, 同一条 TCP 连接上会同时存在多条数据流的传输数据 (如图 7 所示), 为方便后续的修正还原工作, 需要精准划分出加密数据流的音视频分组数据 CDU, 使划分出的每一组加密数据都只包含 1 个 CDU.

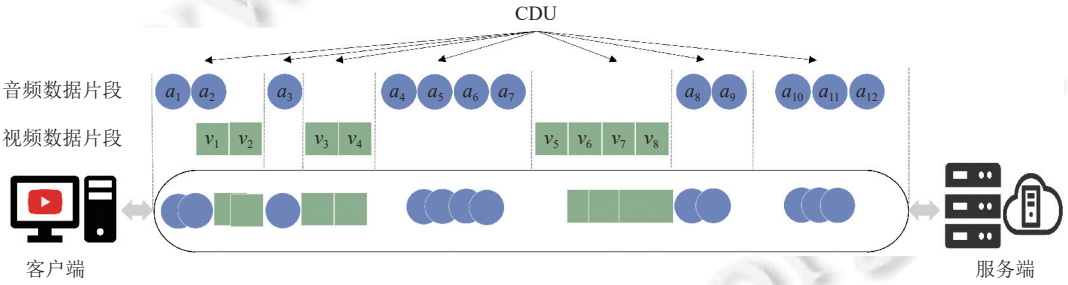

图 7 加密音视频分组数据单元 CDU 精准划分

对 CDU 的划分是通过特定长度的 TLS 记录作为标识. 根据 RFC 7540 标准, WINDOW_UPDATE 帧在 HTTP/2 中用于流量控制. 视频服务器响应各组音视频 CDU 数据时, 会先发送若干个 WINDOW_UPDATE 帧来更新每条 HTPP/2 流的数据窗口大小, 且这些 WINDOW_UPDATE 帧被封装在同一个 TLS 记录中传输. 由于每个 WINDOW_UPDATE 帧大小固定为 13 B[35], 通过观察, WINDOW_UPDATE 帧经 TLS 协议处理封装后, 大小变为 35 B, 包含两个 WINDOW_UPDATE 帧的 TLS 记录大小为 48 B, 正好增加一个 WINDOW_UPDATE 帧的长度, 依此类推, 本文将 35 B、48 B、61 B、74 B 和 84 B 等大小的 TLS 记录作为标志 TLS 记录, 标识 CDU 的开始.

5.3 针对 TLS 协议干扰的修正

公式 (1) 中的参数 TLS_θ_{len} 为单个 TLS 记录在承载传输数据后引入的干扰量的均值, 本节将根据修正还原公式 (1) 构建线性回归修正模型, 利用回归修正模型估算参数 TLS_θ_{len}.

在 HTTP/2 协议中, 音视频 CDU 会被分割后封装在 HTTP/2 DATA 帧中, 然后由 TLS 协议进一步封装到 TLS 记录中, 如果 HTTP/2 DATA 帧的长度超过了 TLS 记录的最大负载长度, 则会被切分到多个 TLS 记录中进行

传输, 但当 HTTP/2 DATA 帧长度较小, 未超过 1 个 TLS 记录的最大负载长度时, 往往只会使用 1 个 TLS 记录传输. 已知 TLS 的最大负载长度为 16 384 字节, 当 HTTP/2 DATA 帧的长度小于 16 384 字节时, 该帧只会使用 1 个 TLS 记录传输.

为了准确地得到 TLS_θ_{len}, 可以只选取数据长度小于 16 384 字节的 CDU 进行模型训练. 该 CDU 数据只会使用一个 HTTP/2 DATA 帧, 且该 DATA 帧也只需一个 TLS 记录传输. 此时, 公式 (1) 中的参数 N_{TLS} 和 N_{H2} 均为 1, 公式 (1) 可以简化为:

$$\widetilde{CDU}_{\text{len}} = TCP\_Payload_{\text{len}} - TLS\_\theta_{\text{len}} - H2\_HEADERS\_Frame_{\text{len}} - H2\_Frame\_Headers_{\text{len}} \tag{2}$$

音频片段的数据量较小, 在本文的研究平台中, 一个音频片段的大小为 14 KB 左右, 远小于 HTTP/2 DATA 帧和 TLS 记录的最大负载长度, 满足上述特殊情况. 此外, 根据第 5.1 节和第 5.2 节的介绍, 公式 (2) 中的 $TCP\_Payload_{\text{len}}$、$H2\_HEADERS\_Frame_{\text{len}}$ 和 $H2\_Frame\_Headers_{\text{len}}$ 等参数可以直接获得. 因而从本研究采集到的约 10 万组 CDU 中提取出满足上述情况的 CDU 的加密传输数据作为训练集, 对公式 (2) 进行拟合. 使用 Facebook 和 Instagram 样本数据集得到的 $TLS\_\theta_{\text{len}} = 22$, 这表明 Facebook 和 Instagram 在使用 TLS1.3 封装 HTTP 数据后, 每个 TLS 记录会使负载数据长度增加 22 个字节.

当使用不同视频平台的数据对公式 (2) 进行训练时, 有可能得到不同的 TLS_θ_{len}. 本文后续使用 Facebook 和 Instagram 训练得到的结果, 代入公式 (1) 就可以对 TLS 协议带来的长度增加干扰进行修正.

5.4 针对 HTTP/2 协议干扰的修正

公式 (1) 中 HTTP/2 协议带来的干扰有两项, 包含的参数有 3 个, 分别为 $H2\_HEADERS\_Frame_{\text{len}}$、$H2\_Frame\_Headers_{\text{len}}$ 和 N_{H2}.

$H2\_HEADERS\_Frame_{\text{len}}$ 为承载 HTTP 首部数据的 HEADERS 帧的长度, 其大小一般小于 TLS 记录的最大报文长度 *MSS*, 所以该帧会单独被封装在一个 TLS 记录中, 并且在传输音视频数据的 DATA 帧之前传输. 该参数的大小可以通过第 5.1 节介绍的方法获得.

$H2\_Frame\_Headers_{\text{len}}$ 是每个 HTTP/2 数据帧的首部长度, 在 HTTP/2 的协议标准 RFC 7540 中被指定为 9 B.

N_{H2} 指承载一个 CDU 的 HTTP/2 DATA 帧的个数. 由于 HTTP/2 数据帧被 TLS 协议加密, 无法通过分组数据获得. 本文利用机器学习模型, 用承载 CDU 数据的 TLS 记录的加密有效负载长度序列作为特征, 所有承载 CDU 数据的 DATA 帧的数量作为标签, 训练机器学习模型. 当需要对一个 CDU 进行修正复原时, 提取承载该 CDU 的 TLS 记录的加密有效负载长度序列, 就可以利用训练出的机器学习模型预测.

上述 N_{H2} 预测模型的训练过程中, 需要提取出每个加密 CDU 对应的 TLS 记录长度序列作为特征. 在第 5.2 节划分出每个加密 CDU, 并提取出每个加密 CDU 对应的多个 TLS 记录长度序列, 在第 5.3 节利用线性回归模型拟合出每个 TLS 记录对其所承载的数据的长度干扰量为+22 B, 因此将序列中每个 TLS 记录长度减 22 后就是每个 TLS 记录内数据的原长度, 这些 TLS 记录数据原长度序列就是本文提取的 TLS 记录长度序列特征.

N_{H2} 预测模型的目标是预测出每个 CDU 的参数 N_{H2}, 因此训练数据需要有 N_{H2} 值作为标签. TLS_Load_{len} 表示 CDU 数据被 HTTP/2 和 TLS 协议封装后的有效载荷长度总和, 有:

$$TLS\_Load_{\text{len}} = TCP\_Payload_{\text{len}} - TLS\_\theta_{\text{len}} \times N_{\text{TLS}} \tag{3}$$

将公式 (3) 代入公式 (1) 可得:

$$\widetilde{CDU}_{\text{len}} = TCP\_Load_{\text{len}} - H2\_HEADERS\_Frame_{\text{len}} - 9 \times N_{H2} \tag{4}$$

从公式 (4) 可得到计算参数 N_{H2} 的公式:

$$N_{H2} = (TLS\_Load_{\text{len}} - H2\_HEADERS\_Frame_{\text{len}} - \widetilde{CDU}_{\text{len}})/9 \tag{5}$$

利用公式 (5) 和 CDU 对应的明文指纹便可以计算出参数 N_{H2} 的值, 将该值作为上述特征序列的标签.

通过上述提取方法, 从样本 CDU 提取出的 TLS 记录长度序列作为特征, 对应的 N_{H2} 作为标签, 就可以训练出预测模型.

为了提高模型预测的准确率, 本文对用于预测模型训练的特征向量做了进一步的处理. 在 HTTP/2 协议和

TLS 协议对数据的封装过程中, 有时会将 DATA 帧数据切分为连续的、等长的 TLS 记录. 为了更好地刻画 CDU 的特征, 并计算 CDU 中包含的 DATA 帧的个数, 属于同一 DATA 帧的多个 TLS 记录应当被合并. 因此, 本文将一条流中出现次数超过阈值的连续、等长的 TLS 记录定义为这条流的频繁项, 然后将训练集中所有 CDU 对应的 TLS 记录序列进行合并频繁项操作. 通过这种合并频繁项的操作, 使得特征向量中包含的特征值更贴近真实 DATA 帧的长度, 并且显著减少了特征向量的长度. 如图 8 所示, 该方法能显著提升模型的准确率.

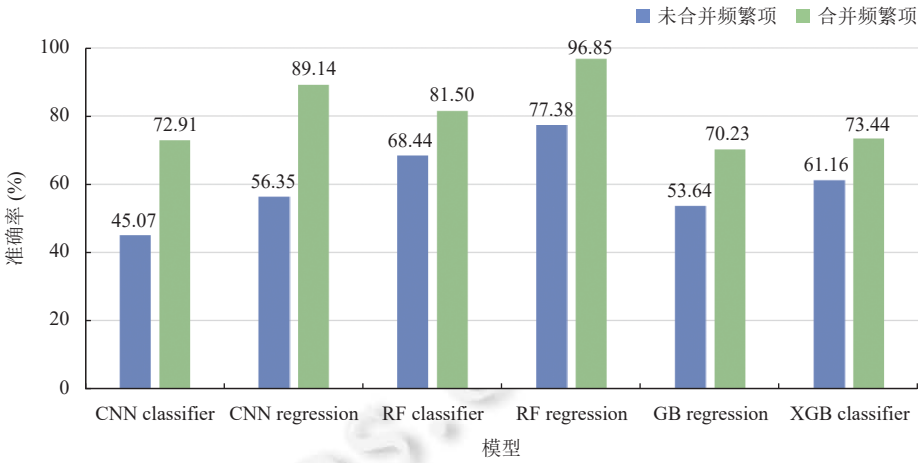

图 8 各机器学习模型的预测效果对比

为选取最适合处理该问题的机器学习模型, 本文选取了几种常用的机器学习模型进行训练. 如图 8 所示, 在对 TLS 负载序列合并频繁项之后, 本文所选取的所有模型的准确率都有更好的表现. 在这些常见的机器学习模型中, 随机森林回归模型 (random forest regression model) 的表现最好, 其对合并频繁项之后的特征序列的预测准确率甚至达到了 96.85%, 且其模型所需训练时间也最少, 因此本文选用随机森林回归模型训练 HTTP/2 的干扰修正模型进行后续实验.

至此, 与 HTTP/2 协议带来的干扰有关的 3 个参数: $H2\_HEADERS\_Frame_{\text{len}}$、$H2\_Frame\_Headers_{\text{len}}$ 和 N_{H2} 都可以获得, 将这 3 个参数代入公式 (1) 就可以对 HTTP/2 协议带来的长度增加干扰进行修正.

5.5 修正结果分析

本文针对 Facebook 和 Instagram 平台共计采集了 7 846 个视频的加密传输数据, 对这些加密视频数据划分组合单元 CDU 后, 与对应视频的明文指纹进行预匹配, 得到 110 434 组匹配的 CDU 的 TLS 序列及其视频标签作为训练和测试数据, 按照 7:3 划分为训练集和测试集, 将训练集送入模型进行训练, 随后使用训练好的模型进行修正还原, 测试集的 33 129 组 CDU 长度还原误差分布如表 2 所示.

表 2 加密音视频数据分组数据单元 CDU 修正还原误差分布

误差 ($\widehat{CDU}_{\text{len}} - CDU_{\text{len}}$) (字节)	数据量	占比 (%)	累计占比 (%)
$\lvert\widehat{CDU}_{\text{len}} - CDU_{\text{len}}\rvert = 0$	27 925	84.291 7	84.291 7
$1 \leqslant \lvert\widehat{CDU}_{\text{len}} - CDU_{\text{len}}\rvert \leqslant 9$	4 785	14.443 5	98.735 2
$10 \leqslant \lvert\widehat{CDU}_{\text{len}} - CDU_{\text{len}}\rvert \leqslant 18$	277	0.836 1	99.591 4
$19 \leqslant \lvert\widehat{CDU}_{\text{len}} - CDU_{\text{len}}\rvert \leqslant 27$	43	0.129 8	99.701 2
$28 \leqslant \lvert\widehat{CDU}_{\text{len}} - CDU_{\text{len}}\rvert$	99	0.298 8	100

由表 2 可知, 本文提出的面向 HTTP/2 协议加密视频修正指纹构建方法对加密视频传输指纹的修正还原误差较小, 在测试集的 33 129 组加密数据中, 修正还原的平均误差为 1.75 B, 标准差为 8.84 B, 其中 84.29% 的加密数据长度实现了 0 误差还原, 超过 99.70% 的数据还原误差在 28 B 以内, 误差分布基本符合正态分布. 以一个包含 4

段 144p 视频片段的 CDU 为例, 其大小约为 56 KB, 28 B 的误差仅占 CDU 实际长度的 0.5‰, 而对于分辨率更高, 包含视频片段数量更多的 CDU 而言, 其数据长度大小往往远大于 56 KB, 因此, 超过 99.70% 的指纹还原误差都在 0.5‰范围内. 由此可见, 本节所提出的加密数据修正还原方法在处理加密视频数据时具有较高的准确性. 这也为后续的视频指纹匹配识别提供了良好的基础.

6 加密视频识别模型

使用第 5 节的方法构建加密视频修正指纹后, 就可以使用修正指纹与明文指纹库进行匹配以实现加密视频识别. 设计匹配算法首先需要分析加密视频修正指纹与明文指纹的关系. 由于使用了多路复用技术, 使用 HTTP/2 协议的客户端在视频播放阶段可能会一次请求连续 n 段相同分辨率的视频片段或音频片段, 而本文定义的 CDU 为服务器对这样的请求的响应数据, 因此本文在第 5 节中修正还原出的加密视频修正指纹分片与视频的明文指纹分片并非一一对应的关系, 一个修正指纹分片可能对应视频明文指纹中的 1 个或者多个连续分片. 除此之外, 在视频实际播放过程中往往会发生视频分辨率自适应切换事件, 这导致了播放视频对应的明文指纹可能是由视频的不同分辨率的明文指纹的多个部分组合而成, 使得一条修正指纹可能与多条明文指纹的不同部分相匹配. 在设计修正指纹序列和视频明文指纹序列的匹配识别算法时, 必须适用以上实际情况, 图 9 为本文提出的加密视频匹配识别算法.

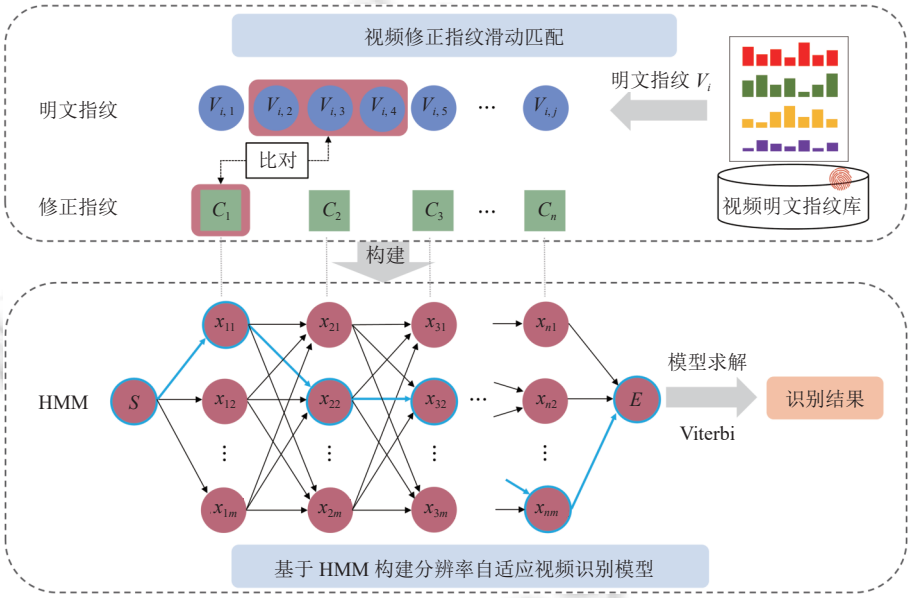

图 9 加密视频匹配识别算法原理

如图 9 所示, 该加密视频识别算法分为两个阶段: 视频修正指纹滑动匹配阶段和视频识别阶段. 在视频修正指纹滑动匹配阶段, 基于滑动窗口思想, 将加密视频修正指纹存入视频明文指纹库中进行匹配, 输出每个修正指纹分片可能的匹配结果. 在视频识别阶段, 利用得到的匹配结果和转移概率矩阵等信息, 构建隐马尔可夫模型, 最后通过维特比 (Viterbi) 算法求解模型, 计算出一条最有可能匹配加密视频修正指纹序列的视频明文指纹序列, 从而达到识别加密视频的目的. 本节给出加密视频识别算法两个阶段的设计原理.

6.1 视频修正指纹滑动匹配

视频修正指纹滑动匹配是指将还原出的修正指纹与明文指纹库中的 1 个或者多个明文指纹分片匹配, 从而得到每个修正指纹分片可能的匹配结果. 由于一个修正指纹分片可能对应连续多个视频明文指纹分片, 本文采用动

态调整滑动窗口大小的方法实现这一匹配过程. 这一过程将得到修正指纹与明文指纹库中各指纹分片组合的匹配结果及其相匹配的概率, 供视频识别模型使用.

图 10 中给出了一个修正指纹分片通过在明文指纹上的滑动匹配得到候选指纹分片组合的原理. 由于音视频片段会组合传输, 因此一个修正指纹分片可能与视频明文指纹库中任一视频的 1 个明文指纹分片或者多个连续分片的组合相匹配. 为了实现修正指纹分片与明文指纹分片一对多的匹配, 本文采用动态调整滑动窗口大小的方法进行匹配. 首先, 将滑动窗口大小 WS (WindowSize) 分别设置为 1–n, 并在视频指纹库的明文指纹上滑动, 其中, n 与视频平台的内容分发机制有关, 需要根据实际测定获得. 在窗口滑动过程中, 窗口 P 内的指纹分片构成一个指纹分片组合与修正指纹分片进行误差范围匹配. 例如, 图 10 中的明文指纹 V_i 包含 8 个明文指纹分片 $S_1,S_2,\ldots,S_8$, 当滑动窗口大小为 3 时, 指纹分片组合为 $S_1+S_2+S_3,S_2+S_3+S_4,\ldots,S_6+S_7+S_8$ 随后, 将修正指纹分片与所有指纹分片组合进行范围匹配. 匹配的误差范围 $[-T, T]$ 需要根据修正指纹分片的修正误差来设定, 本文已在第 5.5 节中给出了修正指纹分片 CDU 长度修正的误差分布, 并计算了其标准差为 8.84, 且其误差分布基本符合正态分布, 因此本文使用 3 倍标准差即 27 作为匹配时的误差阈值上限, 相应的−27 为误差阈值下限, 因而匹配误差范围为 [−27, 27]. 最后, 本文基于匹配误差得到每个被匹配到的指纹分片组合的匹配概率, 由于误差分布基本符合正态分布, 因此匹配误差可以通过标准差为 8.84 的正态分布概率计算公式获得. 通过以上过程, 就可以得到与各修正指纹分片相匹配的明文指纹分片组合及其相匹配的概率.

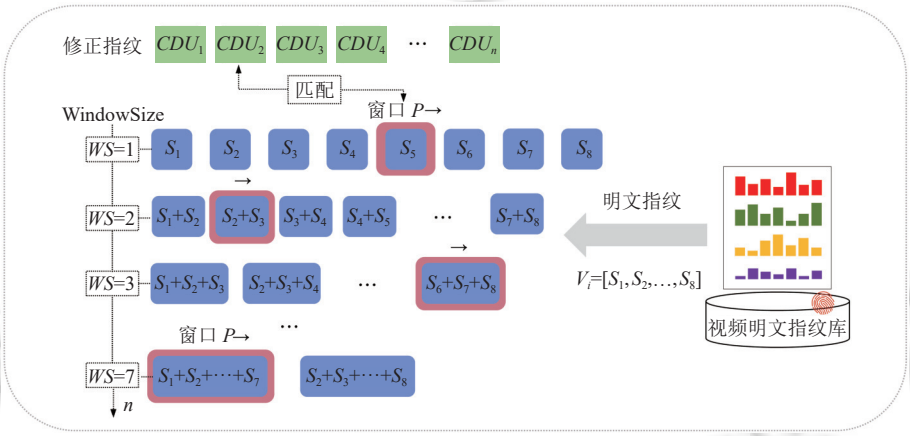

图 10 修正指纹分片与视频明文指纹滑动匹配原理

然而, 随着视频指纹库规模的增大, 修正指纹所需要进行匹配的次数也会增加, 进而导致加密视频的识别时间急剧增加. 若继续使用滑动窗口算法进行指纹匹配, 则难以满足在大规模指纹库场景下的时间开销要求. 因此, 本文在滑动窗口算法的基础上设计了两种优化方法: (1) 设计内存键值对数据库加快匹配速度; (2) 将匹配过程划分为全匹配和快速匹配两个阶段减少匹配次数. 通过这两种优化方法, 能够大大缩短匹配时间, 使得本文方法在大规模指纹库场景下能够实现快速匹配.

(1) 设计内存键值对数据库加快匹配速度

本文根据图 10 中视频明文指纹滑动组合的原理在内存中生成了键值对数据库. 在键值对数据库中, 每个键为一个明文指纹分片组合的长度, 键值为明文指纹分片组合对应的描述信息 (例如视频 ID、 分辨率、 组合片段等), 对于键相同的键值对, 本文使用了链表结构进行存储. 由视频明文指纹生成的键值对数据库的结构如图 11 所示.

在修正指纹匹配过程中, 对于每个修正指纹分片, 根据其大小和匹配误差范围, 在键值对数据库中确定修正指纹分片匹配到的键的范围, 接着遍历匹配到的各个键所对应的链表, 获取明文指纹分片组合匹配结果并计算匹配概率.

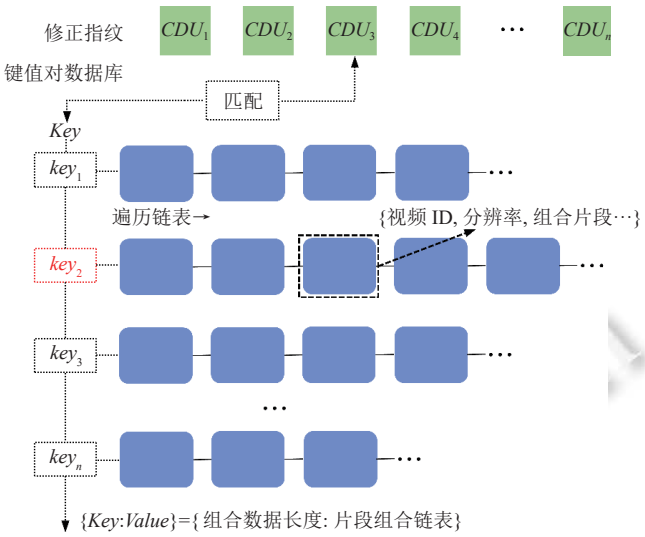

图 11 修正指纹与明文指纹键值对数据库匹配原理

相较于对所有明文指纹分片组合进行匹配的简单滑动窗口算法, 上述使用键值对数据库的滑动匹配算法, 仅对在误差范围内的键所关联的明文指纹分片组合进行匹配, 从而减少了匹配次数, 加快了匹配速度. 除此之外, 为了加快数据的读写速度, 本文将键值对数据库中的指纹数据直接存储在内存中. 该优化方法大大加快了修正指纹匹配的速度.

(2) 将匹配过程分为全匹配和快速匹配两个阶段减少匹配次数

如图 12 所示, 为了进一步缩减指纹匹配时间, 加快指纹匹配速度, 本文将整个修正指纹的匹配过程分为全匹配和快速匹配两个阶段. 其中, 全匹配阶段的目的是使用修正指纹的前几个分片进行匹配, 从而在海量指纹库中筛选出所有可能的候选视频, 并组成一个小型候选视频明文指纹库. 在快速匹配阶段, 只需将修正指纹的剩余分片在该小型候选视频明文指纹库中进行匹配, 即可获取所有修正指纹分片的匹配结果及匹配概率. 通过两阶段的匹配, 可以大大减少匹配指纹的个数, 快速获取各修正指纹分片的匹配结果及匹配概率. 具体过程如下.

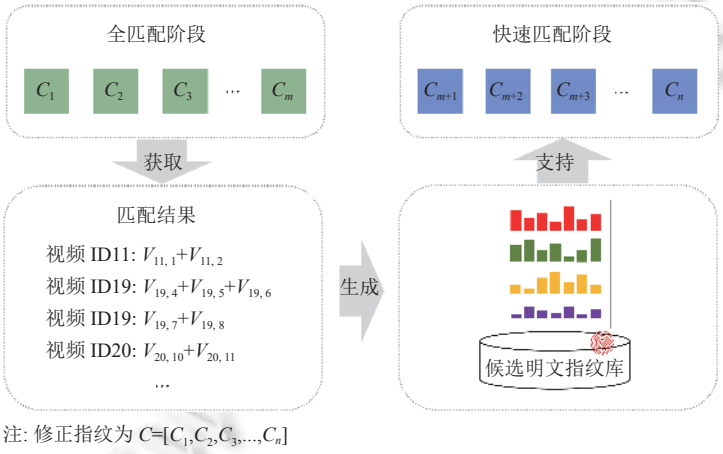

图 12 修正指纹的分阶段匹配过程

全匹配阶段. 首先将修正指纹序列中的前 m 个分片 $C_1,C_2,C_3,...,C_m$ 分别在由大型视频明文指纹库生成的内存键值对数据库中进行匹配, 得到前 m 个修正指纹分片的匹配结果及匹配概率, 然后根据匹配结果中包含的视频

ID, 从大型指纹库中提取出对应的视频明文指纹, 组成一个小型的候选视频明文指纹库.

快速匹配阶段. 将修正指纹的剩余分片 $C_{m+1},C_{m+2},...,C_n$ 在由小型候选视频明文指纹库生成的内存键值对数据库中进行匹配, 从而获取剩余修正指纹分片的匹配结果及匹配概率. 在该阶段, 由于修正指纹分片只在小型的指纹库中进行匹配, 在很大程度上缩小了指纹匹配的规模, 从而实现了快速匹配.

需要注意的是, 图 10 中滑动窗口 P 的窗口上限 n 与视频平台的内容分发机制有关, 需要根据实际测定获得. 为此, 本文分别从 4 个地区采集了 Facebook 和 Instagram 相同视频的加密传输数据, 并结合在代理软件中的观察和与视频明文指纹的对比来对 n 进行设置. 在客户端前期加载视频过程中, 一个 HTTP 请求的连续音视频片段数量 n 较小, 通常为 1–3, 在视频播放的后续过程中, n 会根据网络状况进行调整, 并逐渐趋于稳定. 当网络环境较差时, n 通常为 1 或 2; 当网络环境较好时, n 通常为 6 或 7. 通过对不同地区以及不同分辨率的视频传输数据的分析, 测定 n 的范围为 $1\leqslant n\leqslant 7$. 后续在本节的匹配和识别阶段也证实了上述观察的真实性. 因此本文设置滑动窗口 P 的窗口上限为 7.

6.2 基于 HMM 构建分辨率自适应视频识别模型

通过第 6.1 节的视频修正指纹滑动匹配算法, 获得了各修正指纹分片与视频明文指纹库中视频片段的所有可能的匹配结果, 接下来以音视频片段序列传输顺序和片段间的理论切换概率为准则, 以实际匹配到的音视频序列片段数量和顺序为基础, 构建合适的识别模型, 将识别模型输出的概率最大的识别结果作为最终加密视频内容识别的结果. 根据调研, 常用于语音识别领域的隐马尔可夫模型 (hidden Markov model, HMM)[40]的使用场景与本文的识别场景非常符合, 因此本文选择以 HMM 模型为基础构建视频识别模型, 进而利用维特比算法求解 HMM 模型达到识别视频的目的.

隐马尔可夫模型是马尔可夫链的一个拓展 (如图 13 所示), 隐马尔可夫模型中的任一时刻的状态 S_t 是不可见的, 观察者无法通过观察状态序列 $S_1,S_2,S_3,\ldots,S_t$ 来推测模型中的转移概率等参数, 但在隐马尔可夫模型中, 每个时刻都会输出一个状态 O_t, 且这个状态 O_t 仅和与之对应的状态 S_t 相关, 这也被称为独立输出假设. 在隐马尔可夫模型中, 状态序列 $S_1,S_2,S_3,\ldots,S_t$ 被称为隐含状态序列, 状态序列 $O_1,O_2,O_3,\ldots,O_t$ 被称为输出状态序列或可观测状态序列.

(a) 马尔可夫链 (b) 隐马尔可夫模型

图 13 马尔可夫链和隐马尔可夫模型

基于马尔可夫假设和独立输出假设, 可以计算出某个特定的状态序列 $S_1,S_2,S_3,\ldots,S_t$ 产生输出状态序列 $O_1,O_2,O_3,\ldots,O_t$ 的概率:

$$P(S_1,S_2,S_3,\ldots,O_1,O_2,O_3,\ldots)=\prod_t P(S_t|S_{t-1})\cdot P(O_t|O_{t-1}) \tag{6}$$

解码问题是隐马尔可夫模型的一个主要应用场景. 解码问题是指给定一个模型和某个特定的输出序列, 如何找到最可能产生这个输出的状态序列, 即根据模型和可观测序列寻找隐含状态序列. 解码问题与本研究根据加密视频修正指纹识别加密视频问题非常相似, 从加密视频传输数据中修正还原出的视频修正指纹序列可以看作是隐马尔可夫模型中的输出状态序列, 而视频明文指纹序列可以看作是隐含状态序列, 至于模型中各个状态之间的转移概率也可以根据不同视频片段之间的切换概率计算, 因此, 加密视频识别过程就可以看作根据隐马尔可夫模型和输出状态序列 $O_1,O_2,O_3,\ldots,O_t$ (视频修正指纹序列) 寻找最有可能产生该输出序列的隐含状态序列 $S_1,S_2,S_3,\ldots,S_t$ (视频明文指纹序列), 即:

$$S_1,S_2,S_3,\ldots=\underset{\text{all } S_1,S_2,S_3,\ldots}{\operatorname{argmax}}\ P(S_1,S_2,S_3,\ldots|O_1,O_2,O_3,\ldots) \tag{7}$$

模型的构建需要根据加密视频识别的实际场景计算模型的相关参数. 隐马尔可夫模型主要包括 2 个状态集合和 3 个转移概率矩阵, 共 5 个参数:

$$\lambda = (O, S, A, B, \pi) \tag{8}$$

根据实际视频播放场景的特点和隐马尔可夫模型的需要, 对加密视频识别场景下的隐马尔可夫模型中的参数做出如下定义和计算.

(1) 可观测状态集合 O: $O = \{O_1, O_2, O_3, \ldots, O_N\}$ 为可观测状态序列, 也称输出状态序列, 在加密视频识别的场景中, 为待识别加密视频的修正指纹的各分片的集合, 即每个修正指纹分片为一个输出状态.

(2) 隐含状态集合 S: $S = \{S_1, S_2, S_3, \ldots, S_N\}$ 为隐含状态的集合, 在加密视频识别的场景中, 隐含状态即为加密视频所对应的可能明文指纹分片, 即 S 为通过第 5.1 节动态滑动窗口匹配算法匹配到的各修正指纹分片所对应的各视频明文指纹序列片段的集合.

(3) 状态转移矩阵 A: $A = (a_{ij})$ 为隐含状态概率转移矩阵, 转移概率 $a_{ij} = P(S_j|S_i)$ 表示在隐马尔可夫模型中从前一个状态 S_i 进入当前状态 S_j 的概率为 $P(S_j|S_i)$, 在本文加密视频识别的场景中, 隐含状态为视频明文指纹分片, 隐含状态之间的转移概率即为明文指纹分片对应的各视频片段在视频播放时的切换概率.

由于视频播放遵循一定的连续性, 视频明文指纹序列分片作为隐含状态, 也必然遵循一定的连续性, 因此可以根据视频播放的连续性来设定不同隐含状态之间的转移概率. 例如: 不同视频的明文指纹序列分片之间的转移概率可以设定为 0, 因为通过 TCP 连接下载一个视频时不会在同一连接下载其他视频的片段数据; 在视频播放过程中下载过的视频数据不会再重复下载, 也不会逆序下载, 所以这类分片之间的转移概率也可以设定为 0, 为保证整体联合概率的计算, 本文将转移概率为 0 替换为转移概率为一个极小值; 同一视频的符合传输下载顺序的明文指纹序列分片之间的转移概率可以通过公式 (9) 计算.

$$P(S_t|S_{t-1}) = \begin{cases} 0.6, & S_t \text{和} S_{t-1} \text{属于同一个明文指纹, 且序号连续} \\ 0.2, & S_t \text{和} S_{t-1} \text{属于不同明文指纹, 但序号连续} \\ 0.2 \times \dfrac{n}{N-1}, & S_t \text{和} S_{t-1} \text{序号不连续, 但符合下载时序关系} \end{cases} \tag{9}$$

其中, n 为 S_t 中包含的视频片段的个数, N 为 S_{t-1} 可转移的状态的个数.

(4) 发射概率矩阵 B: $B = (b_{ij})$ 为观测状态输出概率矩阵, 也称发射概率矩阵, 其中, $b_{ij} = P(O_i|S_j)$ 表示在 t 时刻, 隐含状态为 S_j 的条件下, 可观测状态为 O_i 的概率. 在本文识别场景中, 发射概率即为修正指纹分片和明文指纹分片组合的匹配概率.

(5) 初始状态矩阵 π: $\pi = (\pi_i)$ 为初始状态概率矩阵, $\pi_i = P(S_j)$ 表示隐含状态在初始时刻 t=1 时的状态概率为 S_j. 因为本研究主要关注最优隐含状态序列的寻找, 即找到一个加密视频修正指纹序列所对应的视频明文指纹序列, 为了简化计算, 这里将初始状态矩阵 π 设置为常数矩阵, 而主要关注隐含状态之间的转移概率的计算.

有了以上 2 个状态集合和 3 个概率转移矩阵就能构建出隐马尔可夫模型——篱笆网络 (如图 14 所示), 后续可以使用解决隐马尔可夫模型解码问题最常用的维特比算法求解模型, 从而求解出加密视频修正指纹所对应的视频明文指纹序列, 识别加密视频.

本节所构建的加密视频识别模型能够解决视频分辨率自适应切换带来的问题. 这是因为在使用维特比算法的动态规划思想求解隐马尔可夫模型的过程中, 视频分辨率的切换在指纹上表现为在不同明文指纹分片之间的切换, 在隐马尔可夫模型中表现为前后不同隐含状态之间的转移, 因而可将隐马尔可夫模型中的隐含状态看作视频明文指纹分片, 将视频分辨率切换时的明文指纹分片之间的切换看作模型中隐含状态之间的转移, 将明文指纹分片间的切换概率看作隐含状态之间的转移概率, 即可将视频播放过程中的分辨率切换事件完全看作隐马尔可夫模型中的隐含状态的转移事件. 利用维特比算法的动态规划思想在篱笆网络中寻找最短路径的过程, 如图 14 中加粗路径所示, 主要依据转移概率和发射概率在图中寻找联合概率最大的子路径, 在视频指纹层面, 可以理解为在不同明文指纹分片中寻找概率最大的组合指纹的过程, 这个由不同明文指纹分片组成的组合指纹即为分辨率切换的加

密视频数据所对应的明文指纹. 综上所述, 以隐马尔可夫模型和维特比算法为基础的加密视频识别模型很好地解决了视频分辨率切换给视频识别带来的困难.

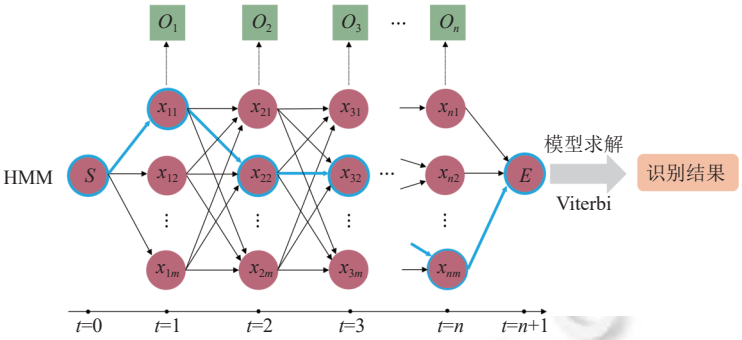

图 14 基于 HMM 识别加密视频

7 加密视频识别实验与结果分析

7.1 实验数据与评估测度

为了全面评估本文方法的实用性, 本文的实验场景包括: (1) 小型指纹库和大型指纹库的实验场景; (2) 固定视频分辨率和自适应视频分辨率的实验场景; (3) 明文指纹和密文指纹的实验场景. 根据以上实验场景的需要, 本文构建了相应实验的数据集. 首先, 本文针对 Facebook 和 Instagram 平台分别采集了 31 万条和 9 万条视频明文指纹, 构建起一个包含 40 万条视频明文指纹的视频明文指纹数据库. 其次, 针对 Facebook 和 Instagram 平台, 本文采集了包含体育、音乐、综艺、动漫、动作、风景这 6 种类型, 时长分布在 90–900 s 之间的视频播放时的加密传输数据, 其数量统计如表 3 所示. 此外, 为满足实验场景 3 中密文指纹识别场景的要求, 本文额外选取了 22 个视频, 每个视频重复采集 40 次, 总计 880 个加密传输数据, 这些数据包含在 Facebook 的自适应分辨率数据集中.

表 3 加密视频传输数据数量统计

平台	类型	播放视频个数
Facebook	固定分辨率	3 150
	自适应分辨率	1 290
Instagram	固定分辨率	2 940
	自适应分辨率	276

考虑到在大型指纹库场景下, 视频识别类别不均衡的问题, 本文使用加权平均方法进行评估测度的计算, 选择准确率 (*Accuracy*)、精确率 (*Precision*)、召回率 (*Recall*)、*F*1 得分 (*F*1_Score) 这几个测度来评估实验的结果. 这些测度的计算都依赖于真阳性 (true positive, TP)、真阴性 (true negative, TN)、假阳性 (false positive, FP)、假阴性 (false negative, FN) 这 4 个参数.

7.2 实验结果与理论分析

(1) 明文视频片段长度作为视频指纹的可行性验证

为了验证使用视频明文指纹在大型视频明文指纹库中识别视频的可行性, 本文对真实视频的片段长度的分布情况进行了研究.

虽然无法获取一个大型视频平台的所有视频片段长度信息, 但根据统计学知识可知, 只要用于研究的样本具有一定的代表性和独立性, 当样本容量足够大时, 就可以使用样本的分布情况评估整体统计情况. 本文所采集的 40 万条视频明文指纹来源于视频平台中真实视频的片段长度信息, 包含体育、音乐、综艺、动漫、动作、风景

这 6 类, 时长为 30–1 800 s 不等. 从 40 万条视频明文指纹中获取了超过 3 000 万个视频片段, 这些片段长度的概率密度函数 (probability density function, PDF) 如图 15 所示.

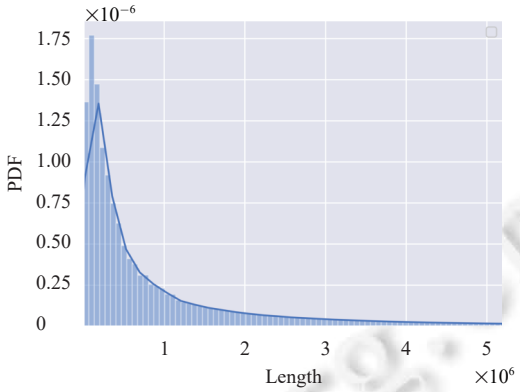

图 15 Facebook 和 Instagram 的视频明文片段长度的概率密度函数

从图 15 中可以获知, 任意两个视频明文片段长度发生冲突的概率最高不超过 1.5×10^{-6}. 假设分段长度发生冲突的概率为 1.5×10^{-6}, 那么对于一个时长为 2 min, 拥有 60 个视频片段的视频而言, 其指纹发生冲突的概率为 $(1.5\times10^{-6})^{60}\approx3.7\times10^{-350}$. 由此可见, 对于一个时长只有 2 min 的短视频, 其视频明文指纹发生冲突的概率就已经非常低了, 如果视频时长更长, 包含更多视频片段, 则视频明文指纹发生冲突的概率会更小. 由此可见, 以视频明文片段长度序列作为视频明文指纹具有很好的唯一性, 从而证明了以视频明文片段长度序列作为视频明文指纹在大型或超大型指纹库中识别视频身份的可行性.

(2) 视频识别方法准确性验证

由于指纹库的规模对视频识别准确率有着较大影响, 为了验证本文视频识别方法的可行性, 本文利用采集的真实加密视频传输数据分别在不同规模指纹库场景进行了准确性测试. 此外, 现有的研究都只验证了在固定分辨率场景下的识别准确性, 但实际播放时, 视频分辨率会随着网络环境的变化发生自适应切换, 这也给识别带来了挑战, 为了验证本方法对真实播放环境的适用性, 本文对固定分辨率和动态分辨率场景都进行了验证. 表 4 为固定分辨率场景下的不同规模指纹库场景中加密视频识别准确性结果, 表 5 为动态分辨率场景下的不同规模指纹库场景中加密视频识别准确性结果.

视频分辨率切换的触发原因包括由网络波动引起的被动分辨率自适应切换和用户手动进行的主动分辨率切换, 本文对 Facebook 和 Instagram 平台分别模拟两种不同情景下的分辨率切换. 为了创建网络状况波动引起的被动分辨率切换场景, 本文在实验采集的局域网环境中使用了软路由设备, 通过该设备利用网络仿真工具 NetEm 和 tc 来控制网络参数. 在 Linux 系统中, 利用仿真工具编写了脚本, 从而可以设置丢包限制网速, 触发被动式分辨率切换.

表 4 固定分辨率场景下的加密视频识别结果

平台	播放视频数量	指纹库数量级 (万)	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
Facebook	3 150	40	98.57	99.67	98.57	99.01
		0.5	99.37	99.57	99.37	99.41
Instagram	2 940	40	98.23	100.00	98.23	99.06
		0.5	99.18	100.00	99.18	99.56
Facebook & Instagram	6 090	40	98.41	99.83	98.41	99.03
		0.5	99.28	99.78	99.28	99.48

表 5 动态分辨率场景下的加密视频识别效果

平台	播放视频数量	指纹库数量级 (万)	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
Facebook	490	40	97.96	97.96	97.96	97.96
		0.5	98.78	98.98	98.78	98.87
Instagram	276	40	97.82	100.00	97.82	98.38
		0.5	98.91	100.00	98.91	99.39
Facebook & Instagram	766	40	97.91	98.69	97.91	98.11
		0.5	98.82	99.35	98.82	99.06

在由网络波动引起的被动分辨率自适应切换实验中, 本文选取了 3 min 左右时长的 Instagram 视频, 在播放过程中通过在中间网关设置网络丢包限制网速的方式改变视频播放的网络环境, 引起视频被动式自适应分辨率切换, 在播放过程中设置 2 次丢包限制, 每次持续 1 min. 在手动进行的主动分辨率切换实验中, 本文选取了视频时长在 3 min 左右的 Facebook 视频, 在播放过程中进行 2 次手动分辨率切换, 每次间隔 1 min. 其实验结果如表 5 所示.

从表 4 和表 5 可以看出, 本研究的方法在不同规模的指纹库识别场景和不同播放模式的识别场景中各项评估指标都表现出了优异的性能, 具体体现在以下两个方面.

(A) 本方法能在大型指纹库中实现较高的识别效果. 从表 4 可以看出, 在固定分辨率场景下, 无论是在 Facebook 平台还是在 Instagram 平台, 本研究所提出的加密视频识别方法在明文指纹数量为 0.5 万的小型指纹库和 40 万的大型指纹库中都实现了较高的识别准确率和精确率, 尤其是在 40 万级的大型指纹库中, 本文识别方法仍实现了超过 98% 的识别准确率和超过 99% 的识别精确率.

(B) 本方法能克服视频自适应分辨率机制给加密视频识别带来的不良影响. 视频播放时的分辨率自适应机制会改变传输数据的流量模式, 给加密视频识别增加一定的难度, 然而本方法使用第 6 节构建的加密视频识别模型, 在极大程度上减少了因视频分辨率切换给加密视频识别带来的影响. 对比表 4 和表 5 的识别效果可知, 本文的识别方法在自适应视频分辨率识别场景中仍有较好的表现, 在 40 万级的指纹库中, 其识别精确率相较于识别固定分辨率视频仅下降 0.5%, 对加密视频识别的准确率仍超过 97.8%.

识别过程中仍然存在一些不准确的情况, 通过分析可知, 主要有两个方面的原因: 其一, 加密视频数据片段的长度不能被 100% 还原, 修正还原出的视频片段长度与其原长度仍存在些许误差; 其二, 由于 HTTP/2 的多路复用和 DASH 的视频传输机制与分辨率自适应机制的作用, 个别情况下视频在播放时并非严格按照视频明文指纹的片段序列传输.

(3) 视频识别方法的通用性和泛化性验证

本文提出的加密视频识别方法结合了领域知识和人工智能方法, 首先使用领域知识对通用传输协议和加密协议分析, 并且由此设计特征提取方法, 然后使用人工智能方法获得各平台的参数. 为了验证本文方法在不同视频平台和环境下的通用性和泛化性, 本文在 Facebook 和 Instagram 的基础上补充选取了 3 个国内外的视频平台: Triller、Twitter 和芒果 TV, 其中, Triller 是一个美国的社交网络和娱乐平台, 以短视频为主; Twitter 是美国知名社交网络及微博客服务平台; 芒果 TV 是国内知名的互联网视频平台.

这 3 个平台同样使用了 HTTP 自适应流媒体技术和 HTTP/2 协议传输音视频数据, 但与 Facebook 和 Instagram 平台不同的是, 这 3 个平台使用的流媒体协议是 HLS 协议, 虽然 HLS 协议和 DASH 协议都属于 HTTP 自适应流媒体技术, 但两者在服务器系统框架和媒体内容分发机制上有着较大的差异, 本文将使用前文介绍的加密视频识别方法, 对这 3 个平台进行方法通用性和泛化性验证.

按照前文所介绍的方法流程, 首先, 采集 3 个平台的测试视频的明文指纹, 并存入大型视频明文指纹库. 接着, 对于各个平台, 分别选取 50 个视频的一次播放数据, 进行平台视频识别模型参数的训练. 然后, 进行大规模测试数据的采集. 最后, 对各平台的测试数据进行视频识别实验.

在固定分辨率播放场景下, 针对 Triller、Twitter 和芒果 TV 这 3 个平台, 实验各选取了 3 184、3 000 和 1 200

个时长范围在 30–300 s 的视频, 采集一次播放数据作为测试数据, 其识别结果如表 6 所示.

表 6 固定分辨率场景下新增平台的识别结果

平台	播放视频数量	指纹库数量级 (万)	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
Triller	3 184	40	99.75	100.00	99.75	99.86
		0.5	99.87	100.00	99.87	99.93
Twitter	3 000	40	99.13	99.05	99.13	99.08
		0.5	99.20	99.07	99.20	99.11
芒果TV	1 200	40	99.42	100.00	99.42	99.68
		0.5	99.67	100.00	99.67	99.82

在动态分辨率播放场景下, 针对 Triller、Twitter 和芒果 TV 这 3 个平台, 实验各选取了 761、500 和 450 个时长在 3 min 左右的视频进行播放. 鉴于多样化的视频分辨率切换的触发原因, 本文设置了与上文中 Facebook 和 Instagram 平台相似的多种实验场景, 同样使用网络仿真工具 NetEm 和 tc 来控制网络参数, 模拟网络波动.

为了模拟真实场景中触发自适应切换的多种方式, 针对这 3 个平台, 实验设置了不同的分辨率切换环境.

● Triller: 为了模拟在真实网络条件下视频播放的自适应切换场景, 选择以 AUTO 分辨率模式进行播放.

● Twitter: 为了模拟在严重网络波动条件下视频播放的自适应切换场景, 进行了 2 次网络波动设置, 设置具体为 1% 的网络丢包率和 400 ms 的网络延迟, 每次网络波动持续时长为 1 min.

● 芒果 TV: 为了模拟正常用户手动切换分辨率的场景, 进行 2 次手动分辨率切换, 每次间隔 1 min.

实验得到的识别结果如表 7 所示.

表 7 动态分辨率场景下新增平台的识别结果

平台	播放视频数量	指纹库数量级 (万)	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
Triller	761	40	99.47	100.00	99.47	99.72
		0.5	99.60	100.00	99.60	99.77
Twitter	500	40	98.40	98.60	98.40	98.49
		0.5	98.80	100	98.80	98.95
芒果TV	450	40	99.11	100.00	99.11	99.47
		0.5	99.33	100.00	99.33	99.60

从表 6 和表 7 可以看出, 使用本文方法, 无论是在固定分辨率播放场景下, 还是在动态分辨率播放场景下, 这 3 个平台均表现出了较好的识别效果. 其中, 在固定分辨率播放场景下, 识别准确率、精确率、召回率和 *F*1 得分均超过了 99.05%, 在动态分辨率播放场景下, 识别准确率等各项评价指标也均超过了 98.40%, 这也证明了本文识别方法的通用性. 在实验过程中, 本文针对各个新增的平台, 都只各选取了 50 个视频的一次播放数据进行平台模型的训练, 训练出的模型可以识别新增平台上其他未在训练集中的视频, 充分说明了本文方法的泛化性.

考虑到在真实网络环境下可能存在网络抖动和拥塞现象, 本文在动态分辨率播放场景中, 针对 Triller 和 Twitter 平台, 分别模拟了正常网络环境和严重网络波动环境, 从表 7 中的识别结果来看, 本文方法不仅能在良好网络环境下实现超过 99% 的识别准确率, 也能在严重网络波动环境下实现超过 98.40% 的识别准确率, 这也充分说明了本文方法在真实网络环境下具有较强的鲁棒性.

(4) 视频识别方法的时间开销评估

加密视频识别方法的时间开销是方法能否被落地应用的关键问题之一. 本文第 6.1 节介绍了快速匹配的关键技术, 可以减少视频识别的时间开销. 以下对视频识别的时间进行实验评估.

为了验证本文方法在不同规模视频明文指纹库场景中的匹配效率, 对于 0.5–40 万不同规模的指纹库, 本文分别使用 800 次视频播放进行了实验, 记录并计算了识别每个视频的平均耗时, 结果如图 16 所示.

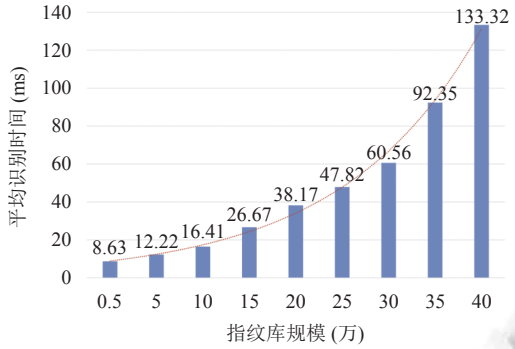

图 16 本文方法在不同规模的指纹库识别场景下的平均匹配识别时间开销

图 16 的结果显示, 虽然随着指纹库规模的增加, 单个视频的平均识别耗时会随之增加, 但从整体角度来看, 本文方法在不同指纹库规模下的时间开销较小, 即使是在 40 万级的指纹库规模下, 单个视频识别时间也仅有 133.32 ms, 这也说明了本文方法能够应用于现实大规模指纹库匹配场景.

8 与现有工作的对比实验与分析

当前加密视频识别研究主要分为两类研究方法, 分别是基于人工智能分类模型的视频识别方法和基于视频指纹的视频识别方法. 由于前者无法识别没有被播放过的视频, 泛化能力很差, 且无法应用于大型视频指纹库场景中, 与本文面向的海量真实场景视频识别应用背景不一致, 因此本文没有与这类研究对比. 后者包括明文指纹识别方法和密文指纹识别方法, 本节对所有现有研究进行了调研分析, 分别选取并复现了这两类方法中的各 1 项最新研究成果, 使用 HTTP/2 传输的视频数据进行了实验, 并与本文的方法进行了比较.

8.1 与现有明文指纹识别方法的对比实验结果与分析

受限于构建大型视频明文指纹库的困难, 大多研究都是基于人工构建的小型视频指纹库进行方法设计和验证. 文献 [22] 基于真实视频指纹模拟构建了 20 万级的视频明文指纹库, 实现了在大规模明文指纹库场景中的加密视频准确识别. 该论文通过分析使用 TLS 协议加密的应用数据单元 (application data unit, ADU) 的密文数据长度相较于明文数据长度发生漂移的原因, 将 TLS 记录特征和 HTTP/1.1 头部特征作为拟合特征来精准还原加密 ADU 长度, 从而构建视频指纹. 在视频识别时, 该方法使用基于序列比对算法进行视频指纹的匹配, 从而识别加密视频. 本文与该方法比较了识别效果和识别时间.

(1) 识别效果比较

本文使用文献 [22] 的方法进行了视频识别对比实验, 使用包含 Facebook 和 Instagram 平台的 HTTP/2 数据集, 分别在固定分辨率和自适应分辨率场景, 指纹数量规模为 0.5 万和 40 万级的视频明文指纹库场景下进行识别实验, 其实验结果如表 8 所示.

表 8 使用明文指纹的对比方法[22]在 HTTP/2 数据集上的识别效果

方法	数据量	分辨率类型	指纹库数量级 (万)	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
本文方法	6090	固定	40	98.41	99.83	98.41	99.03
			0.5	99.28	99.78	99.28	99.48
	766	自适应	40	97.91	98.69	97.91	98.11
			0.5	98.82	99.35	98.82	99.06
对比方法[22]	6090	固定	40	65.35	95.61	65.35	74.77
			0.5	74.91	97.21	74.91	82.59
	766	自适应	40	53.39	79.94	53.39	61.18
			0.5	64.19	89.42	64.19	71.44

对比表 8 与表 4 和表 5 的识别结果, 显然, 在固定分辨率加密视频数据集上, 文献 [22] 的方法在识别准确率、精确率和 *F*1 得分等评价指标上都低于本文方法, 尤其是在 40 万级的指纹库场景下, 其识别准确率仅有 65.35%. 导致其识别效果较差的主要原因是, 该论文中提出的将加密数据长度精准还原为明文数据长度的方法利用了 HTTP/1.1 传输模式, 其难以适用于使用多路复用的 HTTP/2 协议传输数据的视频. 此外, 分析表 8 可以看出, 相较固定分辨率, 该论文的方法在自适应分辨率的识别场景中的识别效果显著下降, 原因是该论文提出的方法并没有考虑视频播放时视频分辨率自适应切换给识别带来的影响, 缺乏鲁棒性.

(2) 识别时间比较

为了评估视频识别方法的时间开销, 本文在相同数据集上进行实验, 得到了文献 [22] 的识别时间, 其结果如表 9 所示.

表 9 使用明文指纹的对比方法[22]在 HTTP/2 数据集上的识别时间统计

平台	识别视频数量	分辨率类型	视频平均时长 (s)	指纹库数量级 (万)	文献[22]方法平均识别时间 (ms)	本文方法平均识别时间 (ms)
Facebook & Instagram	6090	固定	257.19	40 0.5	17150.32 1495.85	245.07 25.66
Facebook & Instagram	766	自适应	183.66	40 0.5	2660.67 184.78	128.02 10.65

从表 9 可以看出, 本文方法相较于文献 [22] 使用的基于序列比对的算法, 无论是在 0.5 万级的指纹库还是在 40 万级的指纹库规模下, 对单个视频的平均识别时间均有数十倍的优势. 通过对表 9 数据的仔细对比, 发现随着指纹库规模的扩大和视频平均时长的增加, 本文方法在时间开销上的优势呈现进一步扩大的趋势.

8.2 与现有密文指纹识别方法的对比实验结果与分析

密文指纹识别方法的指纹库构建相对容易, 因此在该研究方向上已有较多的文献. Bae 等人[25]的研究成果相对较新. 该方法是利用加密视频传输的流量特征作为输入特征, 以 CNN 神经网络模型作为视频标题分类器来达到识别加密视频的目的. 本文从 3 个方面与其进行了比较.

(1) 识别效果与样本容量的关系

本文按照文献 [25] 中的数据集规模, 构建了一个与该论文类似的数据集. 数据集选取了 22 个时长在 2.5 min 左右的 Facebook 视频进行自适应分辨率场景下的播放, 且每个视频重复采集了 40 次. 实验使用不同数量的训练样本进行了训练, 表 10 给出了 Bae 的识别效果与训练集样本容量的关系.

表 10 使用密文指纹的对比方法[25]识别效果与训练集样本容量的关系

实验序号	训练集数据量	测试集数据量	比例	划分方式	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
1	660	220	3:1	随机	87.27	92.16	85.45	88.68
2	586	294	2:1	随机	84.27	89.06	82.52	85.66
3	440	440	1:1	随机	70.00	76.94	67.50	71.91

从表 10 中可以看出, 当模型的训练样本容量变小时, Bae 方法识别的准确率、精确率和召回率等评价指标也随之下降. 这是因为虽然训练集和测试集是在相同环境下采集的, 但在视频播放过程中, 网络波动和服务器状态变化会对加密数据传输的流量模式产生影响, 因此需要尽可能大的样本容量来平衡流量模式突变带来的影响. 由此可知, Bae 方法等密文指纹识别方法的识别效果严重依赖于训练集的样本容量, 需要大量的训练样本来确保识别模型的识别精度. 而本文方法只需要少量样本来训练各平台的修正参数, 精度不依赖训练样本的数目.

(2) 泛化性比较

为了探究 Bae 等密文指纹方法的泛化性能, 本文设置了两组实验, 结果如表 11 所示.

为了探究训练集中样本流量模型的多样性与识别效果之间的关系, 在相同的采集设备和播放网络环境下, 在

表 11 中的实验 1 中, 本文使用了与表 10 中的实验 1 相同的数据集, 并且设置了相同的训练集样本容量. 表 10 中的实验 1 采用随机的方式划分训练集和测试集, 而表 11 中的实验 1 采用分类的方式划分, 分类的依据为采集过程中视频是否发生了分辨率切换从而导致其流量模式发生改变. 将采集过程中流量模式未发生改变的 30 条数据样本作为训练集, 其余作为测试集. 通过这种划分方式, 可以使训练集中的数据样本的流量模式较为单一, 通常为固定分辨率的视频数据样本.

表 11 使用密文指纹的对比方法[25]泛化性研究

实验序号	训练集数据量	测试集数据量	比例	划分方式	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
1	660	220	3:1	分类	68.19	72.73	65.45	68.90
2	880	110	8:1	分类	50.00	56.34	43.48	49.08

从实验结果来看, 表 11 中实验 1 的识别效果远低于表 10 中实验 1 的识别效果, 由此可知, Bae 方法等密文指纹识别方法的识别效果与训练集中样本的流量模式的多样性成正相关, 训练集中的样本越单一, 其识别效果越差. 然而, 在网络易发生波动的环境中, 采用自适应分辨率策略播放的视频很容易发生分辨率的切换, 叠加时间因素后, 一个视频的加密传输数据往往具有多种流量模式, 使用单一流量模式或部分流量模式的数据样本训练出来的识别模型, 对于流量模式不在训练集中的样本往往具有较差的识别效果.

为了进一步研究不同播放网络环境对识别效果的影响. 本文使用相同的采集设备, 在另一个接入环境下, 对每个视频采集了 5 遍加密传输数据作为测试集, 之前采集的 40 遍数据作为训练集训练模型, 模型的识别效果如表 11 中实验 2 所示, 相较于前面的实验, 尽管该实验的训练集的样本容量较大, 但却表现出了更差的识别效果, 究其原因, 是网络环境的改变导致视频播放时传输的承载音视频数据的 CDU 的大小发生了变化, 从而导致整体流量特征发生变化, 进而导致较差的识别效果. 从表 11 的实验结果可见, Bae 方法等密文指纹识别方法的泛化性较差, 难以在不同网络环境中应用.

为了给出本文方法在该数据集上的识别结果, 利用第 5 节训练好的加密数据修正模型和第 6 节的加密视频识别模型对表 10 和表 11 的实验中使用的数据集进行识别, 其在不同数量等级的指纹库中的识别效果如表 12 所示.

表 12 本文方法在该数据集上的识别效果

指纹库数量级 (万)	准确率 (*Accuracy*) (%)	精确率 (*Precision*) (%)	召回率 (*Recall*) (%)	*F*1得分 (*F*1_Score) (%)
40	99.79	100.00	99.79	99.89
0.5	99.90	100.00	99.90	99.95

对比表 10、表 11 和表 12 中的实验结果, 可以看出: 与使用明文指纹方法的本方法相比, 密文指纹识别方法的泛化性较差, 非常容易受到播放视频流量模式变化的影响, 需要通过对待识别视频进行大量重复采集来获取充足的样本, 以平衡因网络环境变化造成的流量模式突变带来的不良影响. 相比而言, 本文方法基于领域知识从加密视频流中提取出不受真实网络波动影响的组合数据单元, 可以稳定识别在不同网络环境下采集的加密视频. 本文的识别模型不需要重复、大量地采集待识别视频的加密传输数据训练模型, 只需利用之前训练好的修正还原模型和一个包含待识别视频明文指纹的明文指纹数据库, 就能很好地识别待识别视频, 具有很好的泛化性.

(3) 识别时间比较

本文对 Bae 方法和本文方法在识别实验中的视频识别时间进行了比较, 结果如表 13 所示. 本文选取了 22 个时长为 150 s 的视频进行识别测试. 对于 Bae 方法, 需要对待识别视频重复播放 30 次, 以获取充足的样本构建训练集, 因此训练数据采集时间为 99 000 s; 对于本文方法, 在明文指纹库构建阶段, 需要采集这 22 个视频的所有分辨率的指纹, 明文指纹库的构建时间为 33 s, 在还原公式求参阶段, 主要包括训练数据采集时间和模型训练时间, 本文方法只需对这 22 个视频采集 1 次即可进行模型训练, 因此还原公式求参时间约为 3 650 s.

从表 13 中的各项操作时间开销对比可以看出, 本文方法相较于使用密文指纹方法的 Bae 的方法, 在视频系统识别时间开销上具有非常明显的优势. 通过分析可知, 其主要原因在于, 使用密文指纹的方法需要多次重复播放待

识别的视频以获取稳定的视频传输特征来训练模型, 且对于所有新视频也都要进行相应的训练. 而本文方法只需要对各个平台的少量数据进行 1 次训练即可获取特征修正参数, 从而对平台中的新视频进行识别.

表 13 与使用密文指纹的对比方法[25]在视频系统识别时间上的对比

Bae方法的视频识别时间开销 (s)		本文方法的视频识别时间开销 (s)	
训练数据采集时间	99 000	明文指纹库构建时间	33
测试数据采集时间	3 300	测试数据采集时间	3 300
模型训练时间	97.11	还原公式求参时间	3 650
模型预测时间	0.38	指纹匹配识别时间	0.11
总计	102 397.49	总计	6 983.11

9 总结与展望

为了解决新型协议 HTTP/2 广泛应用给加密视频识别带来的挑战, 本文提出了一种面向 HTTP/2 流量多路复用特征的加密视频识别方法. 该方法包括两个主要步骤. 第 1 个步骤是将多路复用的 HTTP/2 加密数据精准修正还原为音视频数据单元组合长度的方法, 该修正还原方法可以实现超过 99.70% 的指纹还原误差都在 0.5‰范围以内, 这一高精度的指纹修正还原方法为大型指纹库中精准的视频匹配提供了可能. 第 2 个步骤以加密视频修正指纹和视频明文指纹为基础, 设计了一种分辨率自适应加密视频识别方法, 实验结果表明, 该识别模型在 40 万级的真实大型指纹库中, 无论是固定分辨率场景还是自适应分辨率场景都达到了较高的识别性能, 准确率超过 97%, 精确率超过 99%, 召回率超过 97%, $F1$ 得分超过 99%. 本文的研究和实验都是基于真实的大型视频指纹数据集, 与类似工作的对比也证明了本文方法的实用性.

本文的方法基于网络关键接入点的流量数据分析, 应用时不要求视频平台的协作, 这使得本技术不受视频平台的限制, 可方便地部署应用. 本文构建的视频明文指纹库面向动态真实网络, 只需 1–2 s 就能采集一个视频的所有分辨率的明文指纹, 且所需的网络带宽和存储空间都很小, 可以根据公害视频的认定随时进行扩充更新. 因此, 本文方法具有较高的应用价值. 此外, 本方法无须解密网络应用层视频数据, 只是基于视频传输的流量特征对指纹库中的公害视频进行细粒度的内容识别, 因此本方法在对公害视频进行细粒度管理的同时, 也兼顾了通过加密方式进行端到端数据传输的正常用户的隐私. 本方法实际部署的便利性和对正常用户隐私的兼顾, 使得本方法可以满足精细化网络管理前提下的互联网信息共享需求.

本文方法利用了视频数据传输过程中的协议特征, 将领域知识与机器学习相结合, 实现了 HTTP/2 加密视频数据的精准识别. 目前, 互联网上 HTTP/1.1、HTTP/2、QUIC 和 HTTP/3 协议共存的现象正逐步演变, 未来将以 HTTP/2 和 HTTP/3 为主. 面对复杂的网络环境, 增加方法的通用性, 在保证方法高效实用的基础上, 使用领域知识驱动人工智能进行通用的加密视频识别将是本领域未来研究的方向和热点.

References:

[1] Ericsson Mobility Report. 2023. https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/

[2] Shutsko A. User-generated short video content in social media. A case study of TikTok. In: Proc. of the 12th Int'l Conf. on Social Computing and Social Media. Participation, User Experience, Consumer Experience, and Applications of Social Computing. Copenhagen: Springer, 2020. 108–125. [doi: 10.1007/978-3-030-49576-3_8]

[3] Usage Statistics of Default protocol https for Websites. 2024. https://w3techs.com/technologies/details/ce-httpsdefault

[4] Fu CP, Li Q, Xu K. Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis. In: Proc. of the Network and Distributed System Security Symp. (NDSS 2023). 2023. [doi: 10.14722/ndss.2023.23080]

[5] Fu CP, Li Q, Shen M, Xu K. Realtime robust malicious traffic detection via frequency domain analysis. In: Proc. of the 2021 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2021. 3431–3446. [doi: 10.1145/3460120.3484585]

[6] Ni BL, Peng HW, Chen MH, Zhang SY, Meng GF, Fu JL, Xiang SM, Ling HB. Expanding language-image pretrained models for general video recognition. In: Proc. of the 17th European Conf. on Computer Vision. Tel Aviv: Springer, 2022. 1–18. [doi: 10.1007/978-3-031-19772-7_1]

[7] Kumar K, Shrimankar DD, Singh N. Eratosthenes sieve based key-frame extraction technique for event summarization in videos.

Multimedia Tools and Applications, 2018, 77(6): 7383–7404. [doi: 10.1007/s11042-017-4642-9]

[8] Yang LM, Fu SJ, Luo YC, Shi JY. Markov probability fingerprints: A method for identifying encrypted video traffic. In: Proc. of the 16th Int'l Conf. on Mobility, Sensing and Networking (MSN). Tokyo: IEEE, 2020. 283–290. [doi: 10.1109/MSN50589.2020.00055]

[9] Li F, Chung JW, Claypool M. Silhouette: Identifying YouTube video flows from encrypted traffic. In: Proc. of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video. Amsterdam: ACM, 2018. 19–24. [doi: 10.1145/3210445.3210448]

[10] Shi Y, Biswas S. A deep-learning enabled traffic analysis engine for video source identification. In: Proc. of the 11th Int'l Conf. on Communication Systems & Networks (COMSNETS). Bengaluru: IEEE, 2019. 15–21. [doi: 10.1109/COMSNETS.2019.8711478]

[11] Shi Y, Feng DZ, Cheng Y, Biswas S. A natural language-inspired multilabel video streaming source identification method based on deep neural networks. Signal, Image and Video Processing, 2021, 15(6): 1161–1168. [doi: 10.1007/s11760-020-01844-8]

[12] Kattadige C, Raman A, Thilakarathna K, Lutu A, Perino D. 360NorVic: 360-degree video classification from mobile encrypted video traffic. In: Proc. of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video. Istanbul: ACM, 2021. 58–65. [doi: 10.1145/3458306.3460998]

[13] Shen M, Zhang JP, Xu K, Zhu LH, Liu JC, Du XJ. DeepQoE: Real-time measurement of video QoE from encrypted traffic with deep learning. In: Proc. of the 28th IEEE/ACM Int'l Symp. on Quality of Service (IWQoS). Hangzhou: IEEE, 2020. 1–10. [doi: 10.1109/IWQoS49365.2020.9212897]

[14] Gutterman C, Guo K, Arora S, Gilliland T, Wang XY, Wu L, Katz-Bassett E, Zussman G. Requet: Real-time QoE metric detection for encrypted YouTube traffic. ACM Trans. on Multimedia Computing, Communications, and Applications, 2020, 16(2S): 71. [doi: 10.1145/3394498]

[15] Wu H, Li X, Cheng G, Hu XY. Monitoring video resolution of adaptive encrypted video traffic based on HTTP/2 features. In: Proc. of the IEEE INFOCOM 2021—IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS). Vancouver: IEEE, 2021. 1–6. [doi: 10.1109/INFOCOMWKSHPS51825.2021.9484509]

[16] Wu H, Yu ZH, Cheng G, Guo SY. Identification of encrypted video streaming based on differential fingerprints. In: Proc. of the IEEE INFOCOM 2020—IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS). Toronto: IEEE, 2020. 74–79. [doi: 10.1109/INFOCOMWKSHPS50562.2020.9162914]

[17] Reed A, Kranch M. Identifying HTTPS-protected netflix videos in real-time. In: Proc. of the 7th ACM on Conf. on Data and Application Security and Privacy. Scottsdale: ACM, 2017. 361–368. [doi: 10.1145/3029806.3029821]

[18] Shen M, Liu YT, Zhu LH, Xu K, Du XJ, Guizani N. Optimizing feature selection for efficient encrypted traffic classification: A systematic approach. IEEE Network, 2020, 34(4): 20–27. [doi: 10.1109/MNET.011.1900366]

[19] Dubin R, Dvir A, Pele O, Hadar O. I know what you saw last minute—Encrypted HTTP adaptive video streaming title classification. IEEE Trans. on Information Forensics and Security, 2017, 12(12): 3039–3049. [doi: 10.1109/TIFS.2017.2730819]

[20] Liu YT, Li S, Zhang CW, Sun Y, Liu QY. ITP-KNN: Encrypted video flow identification based on the intermittent traffic pattern of video and K-nearest neighbors classification. In: Proc. of the 20th Int'l Conf. on Computational Science. Amsterdam: Springer, 2020. 279–293. [doi: 10.1007/978-3-030-50417-5_21]

[21] Xu SC, Sen S, Mao ZM. CSI: Inferring mobile ABR video adaptation behavior under HTTPS and QUIC. In: Proc. of the 15th European Conf. on Computer Systems. Heraklion: ACM, 2020. 33. [doi: 10.1145/3342195.3387558]

[22] Wu H, Yu ZH, Cheng G, Hu XY. Encrypted video recognition in large-scale fingerprint database. Ruan Jian Xue Bao/Journal of Software, 2021, 32(10): 3310–3330 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6025.htm [doi: 10.13328/j.cnki.jos.006025]

[23] Gu JX, Wang JL, Yu ZW, Shen KL. Traffic-based side-channel attack in video streaming. IEEE/ACM Trans. on Networking, 2019, 27(3): 972–985. [doi: 10.1109/TNET.2019.2906568]

[24] Afandi W, Bukhari SMAH, Khan MUS, Maqsood T, Khan ASU. Fingerprinting technique for YouTube videos identification in network traffic. IEEE Access, 2022, 10: 76731–76741. [doi: 10.1109/ACCESS.2022.3192458]

[25] Bae S, Son M, Kim D, Park C, Lee J, Son S, Kim Y. Watching the watchers: Practical video identification attack in LTE networks. In: Proc. of the 31st USENIX Security Symp. (USENIX Security 2022). Boston: USENIX Association, 2022. 1307–1324.

[26] Karagkioules T, Concolato C, Tsilimantos D, Valentin S. A comparative case study of HTTP adaptive streaming algorithms in mobile networks. In: Proc. of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video. Taipei: ACM, 2017. 1–6. [doi: 10.1145/3083165.3083170]

[27] ISO/IEC 23009-1: 2014 Information technology—Dynamic adaptive streaming over HTTP (DASH)—Part 1: Media presentation description and segment formats. 2024. https://www.iso.org/standard/65274.html

[28] Durak K, Akcay MN, Erinc YK, Pekel B, Begen AC. Evaluating the performance of Apple's low-latency HLS. In: Proc. of the 22nd

IEEE Int'l Workshop on Multimedia Signal Processing. Tampere: IEEE, 2020. 1–6. [doi: 10.1109/MMSP48831.2020.9287117]

[29] Behravesh R, Rao A, Perez-Ramirez DF, Harutyunyan D, Riggio R, Boman M. Machine learning at the mobile edge: The case of dynamic adaptive streaming over HTTP (DASH). IEEE Trans. on Network and Service Management, 2022, 19(4): 4779–4793. [doi: 10.1109/TNSM.2022.3193856]

[30] Yu L, Tillo T, Xiao JM. QoE-driven dynamic adaptive video streaming strategy with future information. IEEE Trans. on Broadcasting, 2017, 63(3): 523–534. [doi: 10.1109/TBC.2017.2687698]

[31] ISO/IEC 23009–6: 2017 Information technology—Dynamic adaptive streaming over HTTP (DASH)—Part 6: DASH with server push and WebSockets. 2024. https://www.iso.org/standard/71072.html

[32] Comparison of the usage statistics of HTTP/2 for websites. 2024. https://w3techs.com/technologies/comparison/ce-http2

[33] Chaudhary S, Shukla NK, Chakraborty S, Maity M. A dataset for analyzing streaming media performance over HTTP/3 browsers. In: Proc. of the 37th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2024. 78069–78081.

[34] Wijnants M, Marx R, Quax P, Lamotte W. HTTP/2 prioritization and its impact on Web performance. In: Proc. of the 2018 World Wide Web Conf. Lyon: Int'l World Wide Web Conferences Steering Committee, 2018. 1755–1764. [doi: 10.1145/3178876.3186181]

[35] Belshe M, Peon R, Thomson M. Hypertext transfer protocol version 2 (HTTP/2). RFC 7540, Internet Engineering Task Force, 2015. [doi: 10.17487/RFC7540]

[36] Yang LM, Fu SJ, Luo YC, Wang YJ, Zhao WT. A clustering method of encrypted video traffic based on Levenshtein distance. In: Proc. of the 17th Int'l Conf. on Mobility, Sensing and Networking. Exeter: IEEE, 2021. 1–8. [doi: 10.1109/MSN53354.2021.00017]

[37] Schuster R, Shmatikov V, Tromer E. Beauty and the burst: Remote identification of encrypted video streams. In: Proc. of the 26th USENIX Conf. on Security Symp. Vancouver: USENIX Association, 2017. 1357–1374.

[38] ISO/IEC 23009-8: 2022 Information technology—Dynamic adaptive streaming over HTTP (DASH)—Part 8: Session-based DASH operations. 2024. https://www.iso.org/standard/80898.html

[39] Transport Layer Security (tls). 2018. https://datatracker.ietf.org/wg/tls/about/

[40] Zou FT, Yu TD, Xu WL. Encrypted malicious traffic detection based on hidden Markov model. Ruan Jian Xue Bao/Journal of Software, 2022, 33(7): 2683–2698 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6282.htm [doi: 10.13328/j.cnki.jos.006282]

附中文参考文献:

[22] 吴桦, 于振华, 程光, 胡晓艳. 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. http://www.jos.org.cn/1000-9825/6025.htm [doi: 10.13328/j.cnki.jos.006025]

[40] 邹福泰, 俞汤达, 许文亮. 基于隐马尔可夫模型的加密恶意流量检测. 软件学报, 2022, 33(7): 2683–2698. http://www.jos.org.cn/1000-9825/6282.htm [doi: 10.13328/j.cnki.jos.006282]

吴桦(1973－), 女, 博士, 副教授, 博士生导师, CCF 专业会员, 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.

刘嵩涛(1998－), 男, 硕士生, 主要研究领域为机器学习, 加密流量分类.

罗浩(1998－), 男, 硕士生, 主要研究领域为加密流量分析, 加密视频识别.

程光(1973－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 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, APT 攻击检测.

赵士顺(2001－), 男, 硕士生, 主要研究领域为加密流量分析, 加密视频识别.

胡晓艳(1985－), 女, 博士, 副教授, 博士生导师, CCF 专业会员, 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.