Lightweight Fingerprint Attack and Encrypted Traffic Analysis on News Articles

David Hasselquist*†, Martin Lindblom*, and Niklas Carlsson*
*Linköping University, Sweden
†Sectra Communications, Sweden

***Abstract*—The news articles we read online can reveal a lot about us. Privacy aware groups have therefore long pushed for the use of HTTPS (encrypted end-to-end communication). In this paper, we present the design and evaluation of a lightweight framework that can (1) successfully identify individual news articles even when the articles are delivered over encrypted connections, and (2) separate between articles associated with different news websites even when the websites are delivered over the same infrastructure. Our results demonstrate that naive use of HTTPS is not enough to prevent attackers monitoring a user's connections from identifying articles that the user reads on the most popular news website. We also provide insights into what makes some websites more/less resilient to our attack, and we use Twitter data to evaluate the effectiveness of an example attack that in addition incorporates the popularity of individual news articles. We are the first to demonstrate and evaluate the practical effectiveness of this type of attack when applied on modern news websites, and our multi-website-based evaluation provides valuable insights into how websites can best protect themselves against this type of attacks. These insights are important for websites that want to help protect the privacy of their users.**

***Index Terms*—Fingerprinting attack, Encrypted traffic analysis, News articles identification**

I. Introduction

If monitored over time, the news articles that a person reads can easily reveal information about a person and their leanings. With most people today obtaining their news online, many entities are therefore dedicating significant effort trying to learn as much as they can about the users' online activities.

In the privacy literature, as well as in modern media, most attention related to this privacy concern have been given to cookie management, GDPR/CCPA, the information that is revealed to content providers and ad providers, or alternatively the protection achieved when browsing using privacy enhancing technologies (PETs) such as Tor. Much less attention has been given to what an eavesdropper that monitors regular HTTPS traffic may learn. Furthermore, due to their focus on Tor and other PETs, most fingerprinting works have focused on determining what websites are visited rather than what news articles that a user reads on certain websites. Two reasons for this research gap may be that (1) most web traffic was not encrypted until recent years (due to initially slow adoption of HTTPS), causing most to focus on the context of PETs, where the problem of identifying websites presented a sufficiently hard challenge, and (2) it being non-trivial to identify individual news articles from the encrypted connections visible to the eavesdropper.

ISBN 978-3-903176-48-5 © 2022 IFIP

Compared to third-party cookies and third-party providers, which make themselves indirectly visible on the webpages (via objects/cookies), eavesdroppers are by their nature more difficult to expose and protect against. Yet, if given the ability to extract even a small fraction of the news articles from the encrypted traffic, such attacker could pose a significant privacy threat to individuals. For example, a network operator may collect and sell user information for marketing purposes, a government may perform mass surveillance, tracking citizens' online interests and reading habits to suppress or highlight political views, or a corporation/organization may profile their employers/members to further their own agenda. An eavesdropper could also be another user located on the same network or simply sniffing packets in the air. For this reason, it is very important to better understand the capabilities of an entity that can monitor the encrypted web traffic.

To address this question, in this paper, we present and evaluate a methodology for identifying individual news articles (with a high probability) even when (1) the news articles are delivered over encrypted connections using Hypertext Transfer Protocol Secure (HTTPS) and (2) many articles may be delivered over the same infrastructure. To break the confidentiality of encrypted web connections, we design a fingerprinting attack based on a lightweight classifier that requires little training and a proxy-less framework for automatically collecting and labeling encrypted TCP traffic. The attack compromises the confidentiality through a combination of deep packet inspection (DPI), analysis of traffic patterns, and matching of digital fingerprints created for individual news articles (e.g., based on the popularity of news articles that day). Our method is shown to provide high F1 scores, recall, and precision; highlighting the danger of these types of privacy violating attacks. Moreover, we show that the size of the X.509 certificate (encrypted with TLS 1.3) can be used to narrow down the potential news websites that a user may read. This allows an attacker to effectively differentiate between popular websites using the same underlying Content Delivery Network (CDN) and that cannot be separated using IP blocks alone. Finally, we show that only a limited number of news articles per news source need to be fingerprinted each day for an attacker to successfully map many articles to a monitored user.

Most prior work (Section VII) have developed fingerprinting techniques for classification of websites, not for individual

news articles, and most these works focus only on identifying a user visiting landing pages. Furthermore, many of these techniques have become increasingly complex, requiring significant training (e.g., of a GAN [1]). In contrast, we take on the more ambitious target of identifying individual news articles (i.e., internal web pages) within the most popular news website visited by a user, and we achieve this using a very lightweight classifier. Furthermore, most prior fingerprinting works only apply within a closed-world setting [2]. In contrast, we base our evaluation on a real-world scenario and provide tangible insights into the vulnerability of the most popular news websites. To drive our real-world scenario, we use links extracted from Twitter, allowing us to incorporate the relative popularity of both news websites and individual articles into our evaluation and discussion of a possible attack.

Outline: Section II describes how we extract the features used for website and article identification. Section III describes the design of our identification framework. The next two sections evaluate the approach for the 10 most popular news websites observed in our Twitter dataset (Section IV) and provide insights into why the basic attack is more/less successful on certain websites (Section V). Section VI then presents and evaluates an example attack that leverages the popularity skew of the articles read each day. Finally, we describe related work (Section VII) and present our conclusions (Section VIII).

II. Feature extraction

Feature extraction from encrypted connection data of website visits is non-trivial. One reason for this is that all websites are implemented differently, loading a website typically involves downloading a large number of files from many different domains, and the files downloaded (e.g., ads) are often personalized for each user. As part of our work, we have identified and use two simple fingerprinting features extracted from the encrypted traffic: the X.509 certificate size and the size of the main document associated with the article. To calculate these features, we first extract the Transport Layer Security (TLS) chunks from each connection.

A. TLS chunk extraction

TLS is used to secure modern HTTPS connections [3]. Identification and authentication are established using certificates issued to web servers by a Certificate Authority (CA). The certificates help clients verify the identity of a web server. Confidentiality and integrity are established using encryption and a message integrity check with a keyed Message Authentication Code (MAC). As the connections use TLS encryption, the contents are kept secret even if intercepted in transit. In this paper, we assume that these encryption methods are secure and instead base our features on the encrypted sizes.

We define TLS chunks as the collection of TLS record sizes of the application data sent from the server to the client. To extract TLS chunks, we concatenate TCP payload bytes sent from the server to the client into one byte-stream. Then, we extract the TLS version by searching for the byte sequence [$0x16$, $0x03$, m], where $m \in \{0x00, 0x01, 0x02, 0x03\}$. The first byte $0x16$ indicates the start of a TLS record of type "Handshake", and the following two bytes indicate the major and minor TLS version, respectively.

TABLE I
Certificate parameters for the most linked news domains (in order) on Twitter at the time of the example study.

Domain	Certificate size	Certificate index
New York Times	$C_s \in \{5176\}$	$C_i \in \{1, 2\}$
Yahoo	$C_s \in \{5253, 4774\}$	$C_i \in \{2, 4\}$
Fox News	$C_s \in \{2933, 2934, 2935\}$	$C_i \in \{2, 4\}$
MSN	$C_s \in \{5558, 5562\}$	$C_i \in \{0\}$
BBC	$C_s \in \{5390, 5310\}$	$C_i \in \{2, 4\}$
NBC News	$C_s \in \{2772\}$	$C_i \in \{1, 3\}$
Forbes	$C_s \in \{2715, 2720\}$	$C_i \in \{1\}$
Buzzfeed	$C_s \in \{3028\}$	$C_i \in \{1, 4\}$
Reuters	$C_s \in \{6280\}$	$C_i \in \{2, 4\}$
New York Post	$C_s \in \{4563\}$	$C_i \in \{2, 4\}$

After acquiring the TLS version, we begin the extraction of TLS chunks. By searching for the byte sequence [$0x17$, $0x03$, m_a] (where the byte $0x17$ indicates a TLS record of type "Application", and m_a is the minor TLS version previously extracted), we can identify all TLS chunks of interest from the byte stream. The two bytes following these sequences contain the TLS record size. By extracting these sequences and TLS sizes, we can form TLS chunks from a TCP connection. For details of TLS bytes and handshake, we refer to the RFC [3]. Our technique works best if all chunks of a connection are captured. In the following, we assume an attacker with a good vantage point (e.g., at the edges of the network) that can observe all packets associated with a connection.

B. TLS certificate size extraction

Studying the encrypted packet streams when downloading a significant number of articles from each news website, including decrypted example connections, we find that (1) the TLS chunk delivering the certificate can be identified from the TLS chunk sequence and (2) the size and its corresponding chunk index can be used to identify which website is accessed. The certificate size is captured by the size of the largest early chunk, and both the chunk size and index are typically the same when connecting to a given website. Here, we use the certificate size C_s with chunk index C_i as a feature to limit the lookup scope of a TCP connection to a specific web domain.

Table I shows the observed certificate parameters for the 10 news websites for which we observed the most news article links on Twitter for the week 2021-02-01 to 2021-02-06, inclusive. (For the data collection, we extracted all tweets with a link and identified those that point to a news domain. Here we focus only on the top 10.) The indices refer to the TLS chunks sent from the server to the client. When identifying the domain of an encrypted connection, we search for a matching TLS chunk of size C_s at index C_i.

C. Document transfer size extraction

To map an encrypted connection to an individual news article on the identified domain, we first extract the transfer size of the main document using a careful reconstruction process. Here, it is important to note that the main document is

almost always fragmented into and delivered as multiple TLS chunks of varying sizes. While these chunk-size sequences at first may appear somewhat random and the reconstruction process is different for each studied news domains (likely due to server-side implementation differences), we have identified predictable patterns that we leverage to reconstruct the transfer size of the main document (and other documents) delivered.

We have found that the file transfer sizes can be reconstructed for most websites using one of two main approaches: "sequence based" and "anchor based". We next describe these approaches and present good settings for each domain.

The sequence based reconstruction process uses unbroken sequences of TLS chunks of size $D_i \in D$ bytes to form the document transfer size. For some domains, we also include a trailing TLS chunk after the sequence in the transfer size or added additional exceptions (e.g., to allow for a smaller number of chunks breaking the sequence but not the transfer of the file). The anchor based process is fundamentally different, and relies on two anchor TLS chunks to mark the start and the end. Here, one size T_s indicates the start and one size T_e indicates the end. We simply calculate the transfer size by summing the size of the TLS chunks in between the first two anchor chunks (i.e., first occurrence of T_s and T_e). Depending on the domain, the size of T_s and T_e are also included in the transfer size. We next present the domain-specific reconstruction process and parameters that we found worked best for each domain.

New York Times: Sequence based with $D \in \{1395, 1055, 202, 40\}$, including trailing TLS chunk. Occasionally, the sequence began with a chunk of size 1395 bytes and then a chunk not in D. In those cases, we filtered these two chunks.

Yahoo: Sequence based with $D \in \{1395\}$. Sequences were often broken repeatably by chunks of varying sizes. These are included if the breaking sequence length was less than four.

Fox News: Anchor based with $T_s = T_e = 2900$, including trailing TLS and anchor chunks. The anchors T_s and T_e mark the first and last occurrence of the chunk size, respectively.

MSN: Anchor based with $T_s = T_e = 33$. The anchors mark the first and second occurrence of the TLS chunk size, respectively. If T_s and T_e were received back-to-back, the anchors mark the second and third occurrences instead.

BBC: Sequence based with $D \in \{1395\}$, including trail.

NBC News: Anchor based with $T_s \in \{72, 2907\}$ and $T_e \in \{843, 844, 845, 846, 847, 848, 849, 744\}$. If $T_s = 2907$, it is included in the transfer size.

Forbes: Sequence based with $D \in \{1395\}$, including trailing TLS chunk. Occasionally, the sequence was broken by TLS chunks of sizes less than 100 bytes, which were filtered.

Buzzfeed: Sequence based with $D \in \{1395, 1055\}$.

Reuters & New York Post: Anchor based with $T_s = 66$ and $T_e = 26$, or $T_s = T_e = 26$ with anchors marking the first and second occurrence of the TLS chunk size.

III. Identification Framework

Our framework is split into two tracks: one for training and one for news article identification. The training track is run before the identification track to provide a collection of fingerprints that the identification module of the second track uses to identify the most likely articles that a user may be reading. Figure 1 shows an overview of these tracks together with their modules. The data collection and extraction modules are used for both tracks. The main difference is in the third module of each track. For the first track, this module simply creates fingerprints, whereas for the second track, the fingerprint of an observed connection is compared against the example fingerprints previously created during the training phase. The implementation is written in C# targeting .NET Core 3.1 to allow for cross-platform functionality.

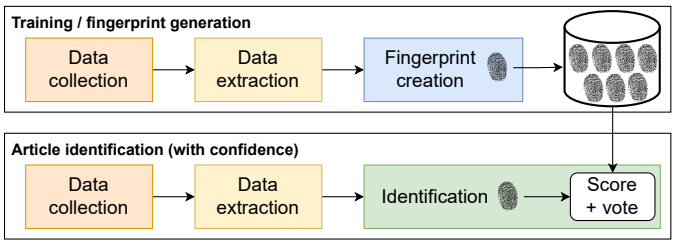

Fig. 1. High-level overview of the system design.

Data collection: The data collection module uses Selenium [4] to browse news websites and capture browsing information. We use SharpPcap [5] for packet capturing and reading/writing to PCAP files. For our experiments, we capture network data while browsing to a predetermined set of URLs, store the actual document transfer size with the server IP address, and create a traffic data object by bundling the browsing information with the PCAP data. Due to our bandwidth limitation (100/100 Mbps), we collect network data from five domains in parallel on five isolated browsers. Both the browser and DNS-cache are cleared between each data collection.

Data extraction: To find the connection transferring the main document, we extract TCP connections and TLS chunks from the PCAP data. Then, we filter the connections and extract those that (1) fall within a time window of when the initial request was made, (2) contain the server IP address, and (3) contain TLS chunks.

Fingerprint creation: From the extracted TCP connections and their TLS chunks, we compare the reconstructed document transfer size to the actual size, and create a fingerprint based on the best matching connection if the difference falls within a maximum allowed threshold. A fingerprint contains the URL, the certificate transfer size, and the document transfer size.

Creating collection of fingerprints: Due to small variations in the transfer size of an individual news article, we collect several fingerprints for each URL of interest. This collection of fingerprints is later used by the identification module to score and classify observed connections. Because of our lightweight design (e.g., only two factors need to be extracted per website visit), the training process requires little time. For example, creating $1,500$ fingerprints from 1.3 million connections takes less than 60 seconds on our 6-core Intel machine (i5-9600K CPU @ 3.70GHz), with 8 GB RAM, running Windows 10 Home. This means that we can quickly

generate new fingerprints and adapt to changes on the web.

Identification: The identification module first extracts the certificate transfer size and matches it against the domain-specific certificate parameters (TLS chunk size and index previously shown in Table I). This allows us to identify the browsed domain and reduce the lookup space. Second, the transfer size of the main document is extracted using our domain-specific reconstruction process.

Finally, we use a voting group system to identify the best matching fingerprint from a set of candidates and use a confidence threshold to decide if we consider the top candidate a good match. This is achieved through the following steps: (1) Each fingerprint (including several fingerprints for the same article) are scored based on the absolute size difference between their and the observed document transfer size. (2) We sort the candidate fingerprints based on their difference score (smallest to largest) and keep the top-V_s, where V_s is the voting group size. (3) We calculate the average score A_i per URL of the n URLs with at least one fingerprint in the top-V_s. (4) For the URL with the lowest average score, say A_i, we calculate a confidence value C as $\prod_{i=2}^{n}\left(1-\frac{A_1}{A_1+A_i}\right)$. (5) Our classification is performed only if the confidence C is above a confidence threshold C_t.

IV. Performance testing

We use single-factor experiments to evaluate the accuracy of our news article identification framework. Specifically, we study the impact of varying one system design parameter at a time, while the other parameters are set to their default values. For each experiment configuration, we run 15 experiments (each with a new set of random pages), and report the average F1 score, recall, and precision metrics. For completeness, we also report the loss rate, defined as the percentage of URLs where the data extraction and fingerprint creation failed. We next describe the factors varied and their default values.

Pages per domain: The number of pages randomly sampled from the data set. $P_d \in \{1, 10, 25, 50, 75, 100\}$, default 10.

Time window: The maximum time threshold for a TCP connection to start after an initial request (used in data extraction). $T_w \in \{10, 100, 500, 1000, 100000\}$, default $500ms$.

Score deviation: The maximum document transfer size difference from the actual size to be considered valid (used in fingerprint creation). $S_d \in \{0, 10, 50, 100\}$, default 10%.

Voting group size: The number of top candidate fingerprints to consider in the voting group system (used in identification). $V_s \in \{1, 10, 5, 20, 50, 100\}$, default 10.

Confidence threshold: The minimum level of confidence to classify a connection as a match (used in identification). $C_t \in \{0, 25, 50, 60, 70, 80, 90, 95, 100\}$, default 50%.

Score threshold: The maximum matching score allowed to classify a connection as a match (used in identification). $S_t \in \{100, 5000, 10000, 100000\}$, default 5000.

A. Extraction Performance

Let us first consider the impact of the three factors used during fingerprint extraction and creation: the number of pages per domain, the time window, and the score deviation. Figure 2 summarizes these results as a 12-by-12 grid. Each row shows the results of a domain (or a collection of domains as in the case with *All* and *Top Performing*) and contains 12 columns grouped into three groups of four. Columns 1-4 show the performance results when varying *Pages per Domain*. Similarly, columns 5-8 and 9-12 show the results when varying the *Time Window* and *Score Deviation*, respectively. In each case, we show the F1 score, recall, precision, and loss rate as a function of the factor. Here, the solid blue and orange lines show the maximum and minimum values, respectively. The dashed blue line shows the mean value for the optimistic setting, and the dashed orange line for the conservative setting. With the optimistic settings, we only include the $P_d \leq 100$ articles used for training in the evaluation, whereas for the conservative setting, we also included the $100 - P_d$ articles not used for the training. As a reference point, we list the average score (over the optimistic and conservative means) achieved using our default configuration (value shown in each plot) and show its default value (vertical dashed line).

Finally, we use shaded regions to show the parameter region over which each metric is within ± 0.2 of the score achieved with the default configuration. In general, the wider this region is, the less sensitive the result are to the parameter value. To emphasize the differences, we color each region based on the size of this region (blue indicates 100% coverage, yellow 0%, and we use linear interpolation for values in between).

In general, we have found the attack to be highly successful for several domains (e.g., F1 scores higher than 0.8 for 7 out of 10 domains, and higher than 0.92 for 4 out of 10 domains), indicating that the method works well for identification of news articles on these popular websites. Furthermore, the results are not sensitive to the selection of the time window or score deviation thresholds (e.g., see large blue regions). With the success rate for several news websites being more sensitive to the number of articles per domain (smaller shaded regions), as discussed later in the paper, it is important to note that most reads each day are associated with a small subset of articles on each website. We next discuss the attack's performance (and each factor's impact) when applied on each domain.

BBC: Our attack performs well on BBC with an F1 score of 0.924, recall of 0.966, and precision of 0.901 at the default configuration. With an increase in the number of pages, the metrics drop to 0.726, 0.653, and 0.817, respectively. The stability is high for the F1 score and precision while being around 50% for the recall. The results show that our attack scales well on BBC with the number of internal pages. For different time windows, the best performance is achieved at or above the default value of $500ms$, while the score deviation performs well at or above its default value of 10%.

Buzzfeed: The performance starts well on Buzzfeed but decrease rapidly with more pages. This is also seen by the small stability region for F1 and recall. In contrast, the precision is near 1.0 for all number of internal pages. The time window and score deviation factors have small impact on the performance if chosen at or above $100ms$ and 10%.

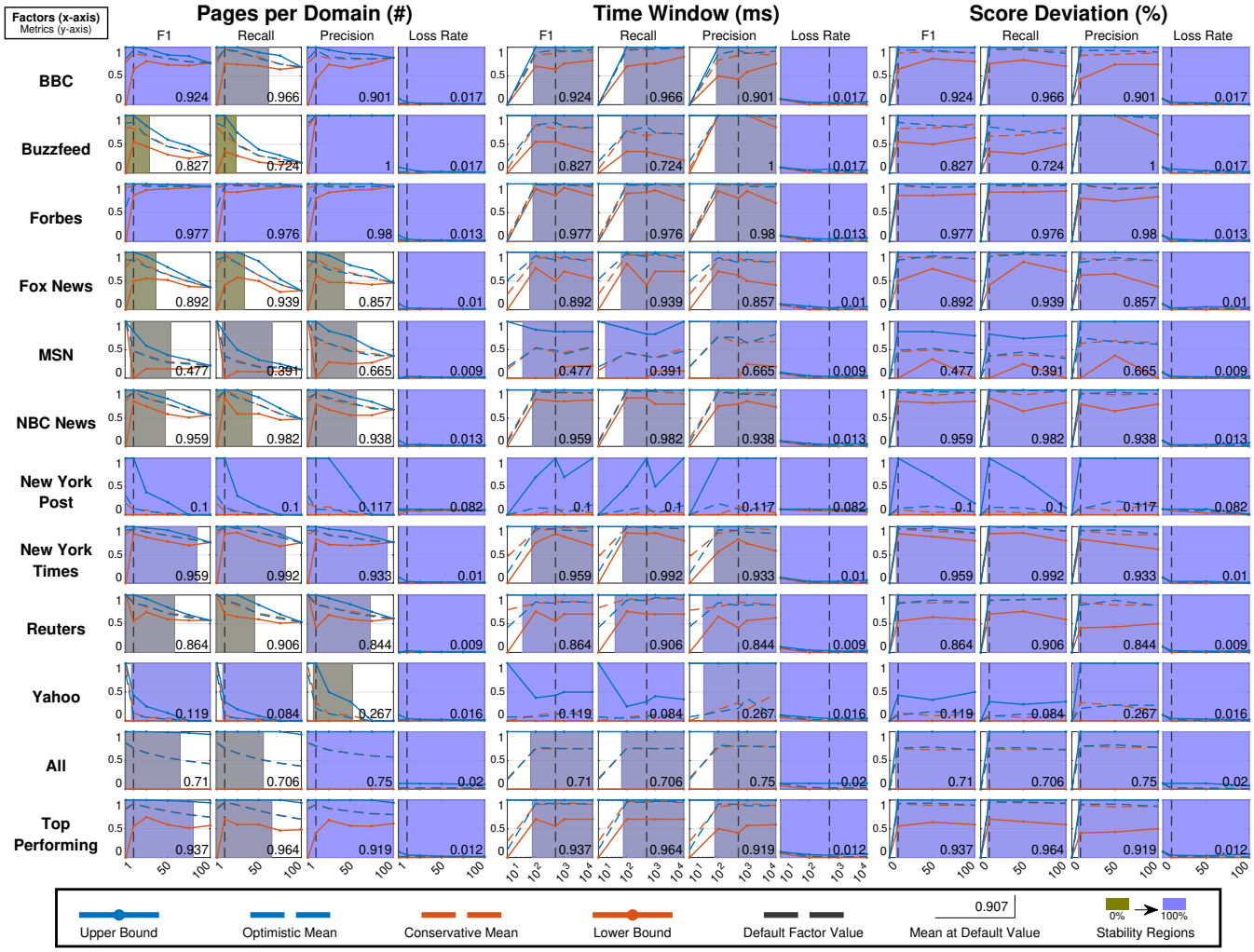

Fig. 2. Performance impact of three fingerprinting parameters (Pages per Domain, Time Window and Score Deviation) measured using four metrics (F1 score, Recall, Precision, Loss Rate) for each of the domains, as well as for all domains and the top performing domains collectively.

Forbes: The attack on Forbes performs well with high stability and mean values for F1 score, recall, and precision (mean values always above 0.8). We find the optimal value for the time window to be at or above $100ms$, and that the score deviation performs well at or above 10%.

Fox News: The performance of Fox News starts well but decreases with the number of internal pages, also shown by the smaller stability regions. The time window and score deviation have limited impact if chosen at or above $100ms$ and 10%.

MSN: The performance on MSN decreases rapidly with the number of internal pages. The results are relatively insensitive around the default time window selection. The choice of score deviation has little-to-small impact if chosen at or above 10%.

NBC News: The performance of NBC News is similar but slightly better than that of Fox News.

New York Post: The performance on New York Post is poor, with low scores for all three metrics, and across the parameter region. The main reason for the poor performance is that New York Post does not have a clear chunk size pattern, making it difficult to extract TLS chunks and identify transfer sizes from the network trace. As we discuss later, this observation may be used as a fingerprinting prevention strategy to help protect against attacks similar to what we demonstrate here. As such, New York Post may serve as an example use case for how fingerprinting can be made more difficult.

New York Times: The performance on New York Times is similar to that of BBC and Forbes, with high scores for all number of pages and large stability regions. The choice of the time window has only a small impact if chosen at or above $100ms$, while the choice of score deviation peaks near 10%.

Reuters: Reuters performs similar to NBC News.

Yahoo: Yahoo performs similar to the New York Post.

All: In general, the number of pages per domain has the largest impact on the F1, recall, and precision. We observe a performance drop as the number of internal pages increases. For example, when combining the results from all domains, the performance starts relatively high but decreases to near 0.5 for all three metrics. The impact of the time window and score deviation is small (e.g., big stability regions), especially if chosen to be at or above $100ms$ and 10%. None of the factors have any significant impact on the loss rate. The small losses are instead due to malformed network traces.

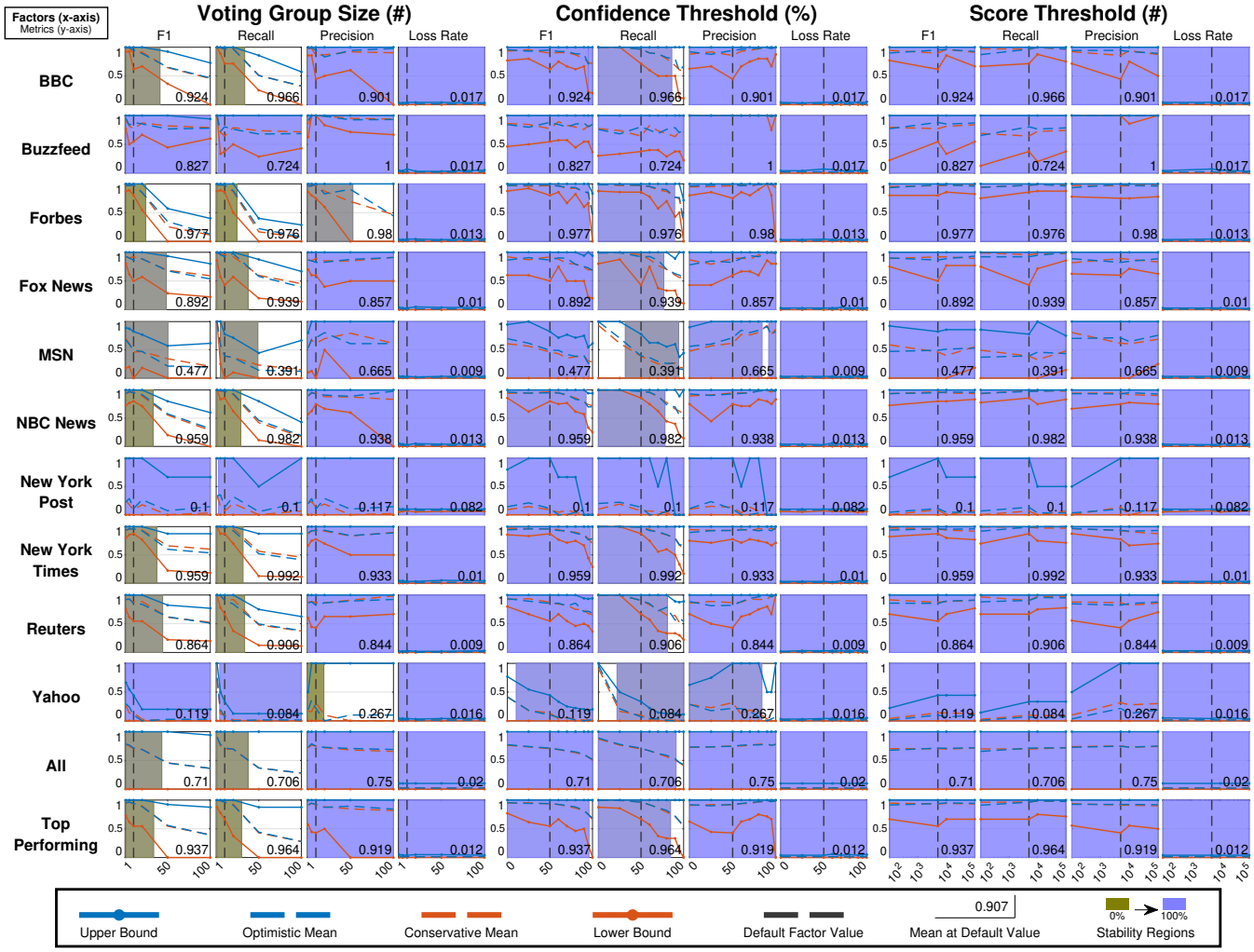

Fig. 3. Performance impact of three identification parameters (Voting Group Size, Confidence Threshold and Score Threshold) measured using four metrics (F1 score, Recall, Precision, Loss Rate) for each of the domains, as well as for all domains and the top performing domains collectively.

Top Performing: Finally, as an attack is expected to be used on domains with a high success rate, we group the top performing domains defined as those with F1 ≥ 0.5 at 100 internal pages (BBC, Forbes, NBC News, New York Times, Reuters). While the values of the three scoring metrics again decrease as the number of internal domains increases, this happens less rapidly compared to for all domains. The time window and score deviation threshold follow a similar pattern.

B. Identification Performance

The three factors used for identification also affect the performance. Figure 3 shows the results when varying the *Voting Group Size*, *Confidence Threshold*, and *Score Threshold*.

In general, the results are not particularly sensitive to the confidence threshold or the score threshold. For these factors, almost all websites have large (blue shaded) stability regions. While the regions are smaller for the voting group size factor, the region is still substantial, making it easy to make a good global threshold choice or selecting domain specific thresholds. We next discuss the results on a per-domain basis.

BBC: We observe a small tradeoff between F1/recall and precision when varying the *Voting Group Size*. The F1 score and recall decrease as the voting group increases, while the precision increases slightly. The stability region for F1 and recall are smaller than for precision, showing that the tradeoff is not equal in absolute values. A similar tradeoff (but with lower scores) is present when varying the confidence threshold. The score threshold has only small impact.

Buzzfeed: The performance metrics are stable on Buzzfeed with large stability regions. For the group size, we observe only a slight decrease when the factor increases. We observe no clear trend when choosing different confidence and score thresholds. Similar to the case with extraction factors, the precision value stays near 1.0 for all identification factors.

Forbes: The attack does not perform well on Forbes with large group size. With small stability regions for F1 and recall, it is clear that a smaller voting group is beneficial on Forbes. For the confidence threshold, we observe a tradeoff between F1/recall and precision. However, the tradeoff is not equal, and an overall performance increase is gained by using a lower confidence threshold. The score threshold has minimal impact.

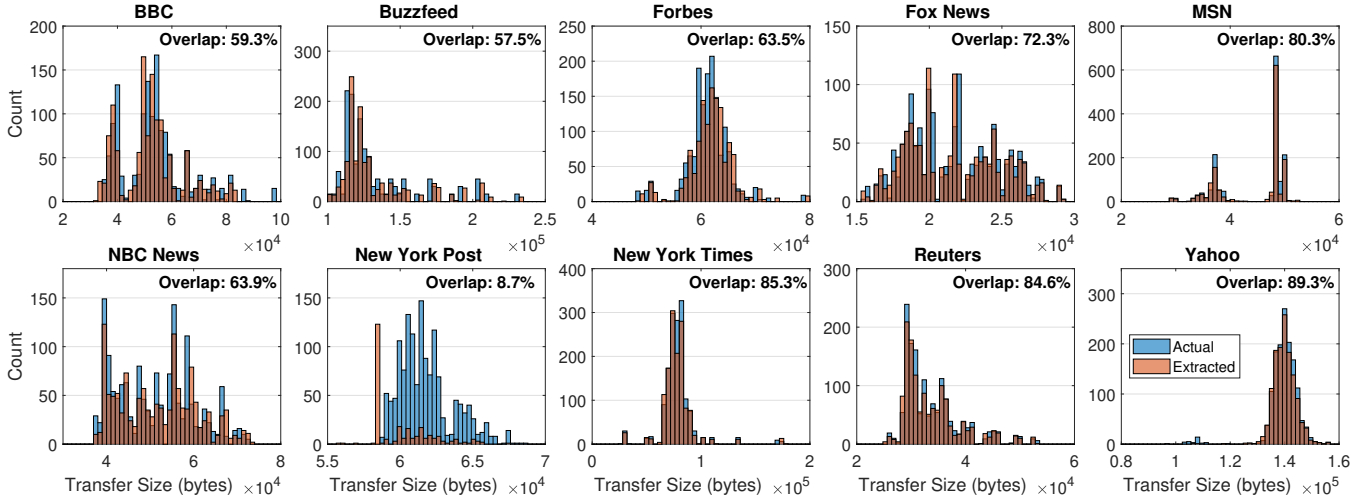

Fig. 4. Comparison of the actual (blue) and extracted (brown) transfer size distributions of each domain, obtained using the default setting and 100 pages per domain. The x-axis scale is selected to encompass both the smallest and largest observed transfer size, and the x-range is split into 50 equally large buckets.

Fox News: The performance on Fox News decreases as the group size increases, with a slight increase in precision. Similar to Forbes, the tradeoff between F1/recall and precision is clearly visible when varying the confidence threshold. Except for small score threshold, this factor has small impact.

MSN: Low and decreasing F1 score and recall are observed even for small voting groups and confidence thresholds. The precision can be increased by selecting a higher confidence threshold. For the score threshold, we observe limited impact.

NBC News: The F1 score decreases significantly with group sizes beyond the default size. The tradeoff between F1/recall is clearly shown for the confidence threshold, and the choice of score threshold has only limited impact.

New York Post: In addition to poor performance, we observe no clear trend when varying the identification factors for New York Post. This again captures that it was difficult to identify and extract the document sizes (from the encrypted traffic) when visiting news articles on New York Post.

New York Times: With small stability regions for F1 score and recall, and only a slight increase in precision, a large group size would be of little benefit here. On the contrary, we observe large stability regions when varying the confidence threshold. Due to a significant decrease in F1/recall when increasing the confidence threshold, there are only small gains with a large value unless aiming for a precision value near 1.0. We observe little-to-no impact for different score thresholds.

Reuters: For Reuters, we again observe a clear tradeoff in F1/recall and precision as the group size and confidence threshold increases, showing some benefits of using larger values. The score threshold has small but visible impact.

Yahoo: Similar to New York Post, we observe no clear trends and poor performance for Yahoo.

All: In general, there is no significant performance gain in increasing the group size, where a size near 10 performs well in most cases. Using a large group size results in a significant drop in F1/recall, but only a small gain in precision. The tradeoff between F1/recall and precision is most visible when varying the confidence threshold. We receive high precision but low F1/recall for high values, while low values result in high F1/recall at the cost of precision. The confidence threshold can thus be tuned based on the desired tradeoff. Furthermore, as the voting group system performs well, varying the score threshold by filtering classifications based on their distance to a fingerprint does not significantly impact performance. The score threshold has little impact in raising the lower bound for the metrics. Similar to the extraction factors, none of the identification factors have any significant performance impact on the loss rate. This is expected as these factors are not connected to the data extraction process. The small deviations are due to random sampling.

Top Performing: We observe similar but better results when considering only the five top performing domains when varying the group size and score threshold. We also receive a significant increase in precision when increasing the confidence threshold. Here, the decrease in F1 and recall might be acceptable in case a high precision is desired.

V. Transfer Size Analysis

Most differences in how well the attack worked on different websites can be explained by the variation in the observed (and actual) transfer sizes of individual news articles. Figure 4 shows the distribution of actual and reconstructed/extracted transfer sizes. With our reconstruction process working well for all domains except New York Post, it is perhaps not surprising that we observe high overlap for all domains except for the New York Post. While there may be ways to accurately extract transfer sizes from the encrypted data for New York Post, the unpredictable nature of the chunking suggest that identification of patterns may be difficult. In fact, we do not rule out the possibility that the patterns we observe is due to a privacy-aware chunking implementation. Based on our findings, potential countermeasures could include websites introducing randomness in the chunking and CAs issuing certificates only from a limited set of "standard" sizes (to increase overlap probability in certificate sizes).

TABLE II
TRANSFER SIZE VARIATION FOR NEWS ARTICLES (BYTES).

Domain	Mean variance	Max variance	Min variance
BBC	1,065	19,469	15
Buzzfeed	53	226	2
Forbes	2,283	22,333	3
Fox News	317	2,239	79
MSN	2,006	26,647	79
NBC News	588	2,499	0
New York Post	1,446	3,956	76
New York Times	1,399	55,820	264
Reuters	606	3,018	46
Yahoo	47,105	144,209	4,992

The relatively poor performance of the other two domains with F1 scores below 0.8 (i.e., MSN and Yahoo) can be explained by two factors that together complicates the identification of articles. First, most articles of these two domains have similar size (e.g., both distributions have one or two clear spikes). Second, the two domains have among the highest relative variation in the document sizes of individual articles. (This is seen by comparing the individual variations shown in Table II with the order of magnitudes observed in Figure 4.) Combined, these two aspects result in dense clustering of fingerprints (with most documents of similar size) and the need to leave room for some flexibility (due to high variance); making it difficult to make high-confidence predictions. We also note that even the slight split into two spikes (MSN) compared to one spike (Yahoo) and relatively less individual variations (MSN relatively less variation than Yahoo) allowed MSN to achieve higher F1 scores than Yahoo.

Finally, for 7 out of the 10 domains the attack is successful (e.g., achieve F1 scores higher than 0.8). The high success rate for these domains can be explained by (1) good matching and (2) sufficiently higher variance in the sizes observed between different articles compared to for an individual article.

VI. DISCUSSION: EXAMPLE ATTACK

While there are many ways to implement and execute the presented attack, we use our Twitter data to discuss the effectiveness of one example approach. First, we note that the reads at any given news website are heavily skewed [6] and that the news cycle typically changes daily [7]. For example, in our dataset, the 10 most popular links of a domain (at a particular day) were responsible for on average 37% of the retweets, the top-50 for 67%, and the top-100 for 78%. Motivated by high correlation between the number of retweets and reads [6], we use the retweets to estimate the fraction of article visits correctly identified by an attacker that track users accessing these websites (e.g., after visiting Twitter; easy to capture using IP address info, DNS data, or a combination thereof) and only use fingerprints for the top-K news articles observed for that website on a given day.

Table III provides conservative lower bound estimates of the expected recall, precision, and F1 scores when using the fingerprints of the top-10 and top-50 articles of each website. For these estimates, we have used the conservative average results of the precision P_K and recall R_K when using only this top-K set and then noted that (1) the recall rate R on the full set of articles observed is the same as R_K and (2) the precision can be lower bounded by $P_{LB} = q_K P_K$, where q_K is the fraction of requests that are to the top-K articles (estimated as the number of retweets of these news articles). This can be easily derived by keeping track of true positive rates and making the conservative assumption that we see the same false positive rate for the pages that are not fingerprinted as those that are fingerprinted. In general, we do better. Finally, we estimate the F1 scores as $F1_{LB} = 2R_K q_K P_K/(R_K+q_K P_K)$.

TABLE III
CONSERVATIVE LOWER-BOUND ESTIMATES OF THE RECALL AND PRECISION WHEN USING THE TOP-K ARTICLES OF EACH SERVICE.

	K=10			K=50		
Domain	R	P_{LB}	$F1_{LB}$	R	P_{LB}	$F1_{LB}$
BBC	0.97	0.48	0.64	0.83	0.61	0.70
Buzzfeed	0.64	0.34	0.44	0.30	0.72	0.43
Forbes	0.98	0.38	0.54	0.96	0.63	0.76
Fox News	0.96	0.41	0.58	0.60	0.49	0.54
MSN	0.39	0.10	0.15	0.21	0.29	0.24
NBC News	0.99	0.36	0.52	0.73	0.56	0.63
New York Post	0.07	0.06	0.06	0.00	0.00	0.00
New York Times	0.99	0.33	0.49	0.89	0.51	0.65
Reuters	0.91	0.27	0.42	0.68	0.37	0.48
Yahoo	0.10	0.10	0.10	0.03	0.05	0.04

Even with these conservative estimates, the attacker could achieve an F1 score above 0.5 for half of the websites. Depending on the long-term objective, we can use $K = 10$ to increase the recall or $K = 50$ to increase the precision. For example, a patient attacker can use the top-10 articles each day to ensure a recall above 0.9 for six of the websites. This shows that an attacker monitoring the service for an extended time period could accurately identify many articles that a user reads even though the traffic is encrypted. From these articles, the attacker can then identify biases, preferences, and other aspects that may compromise the user's privacy.

VII. RELATED WORK

While encrypted traffic analysis is a well-studied area, many works have focused on website fingerprinting [8]–[12] rather than determining what article a user reads on a particular website. Much of this research apply various machine learning techniques [1], [13]–[25] to fingerprint websites obtained using privacy enhancing technologies such as Tor, JAP, OpenSSL, or OpenVPN. Others have proposed various countermeasures [26]–[32] or tried to automate the feature extraction [33], [34]. None of these works try to fingerprint individual news articles, and the only other work that we find (and that claims to be the first) that consider internal pages uses a heavy GAN model [1]. In contrast, our approach is much more lightweight and yet shown to be successful for a practical attack scenario.

Besides identifying browsed web pages, fingerprinting attacks and encrypted traffic analysis can also be used to reveal other information, such as user actions [35], buffer conditions [36], video streams [37], mobile applications [38], voice traffic [39], and messaging applications [40]. For example, Bahramali et al. [40] identify group members of instant messaging applications by studying packet timing and sizes.

Finally, Aqeel et al. [41] highlight the importance of measurement studies to differentiate between landing and internal pages. They show that around two-thirds of the reviewed publications in top-tier networking conferences are not directly applicable to internal pages. Similarly, in contrast to prior fingerprinting research, we focus specifically on the internal webpages (i.e., the news articles) of a few targeted domains.

VIII. Conclusion

In this paper, we have presented the design and evaluation of a framework that can successfully identify individual news articles even when the articles are delivered over encrypted connections. This shows that HTTPS is not enough to protect the user's privacy when visiting a news website. We have also shown that we can separate between articles associated with different news websites even when the websites are delivered over the same infrastructure. This significantly increases the attacker's capability in scenarios where websites end up using the same CDN. By comparing the success rates and the underlying reasons behind these success rates for different websites, we also provide insights into what makes some websites more/less resilient to our attack. Finally, using Twitter data, we have demonstrated the effectiveness of an example attack that in addition incorporates the popularity of individual news articles and discussed tradeoffs associated with how many fingerprints to be used each day. The insights provided are valuable for websites wanting to protect the privacy of their users. Interesting future work include validating the methodology on other internal documents than news articles.

Acknowledgment

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] S. E. Oh, N. Mathews, M. S. Rahman, M. Wright, and N. Hopper, "GANDaLF: GAN for data-limited fingerprinting," in *Proc. PETS*, 2021.
[2] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proc. ACM CCS*, 2014.
[3] E. Rescorla and T. Dierks, "The transport layer security (TLS) protocol version 1.3," RFC 8446, 2018.
[4] Selenium, https://www.selenium.dev/, 2022.
[5] SharpPcap, https://github.com/chmorgan/sharppcap, 2022.
[6] J. Holmström *et al.*, "Do we read what we share? analyzing the click dynamic of news articles shared on Twitter," in *Proc. IEEE/ACM ASONAM*, 2019.
[7] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proc. ACM KDD*, 2009.
[8] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proc. IEEE S&P*, 2002.
[9] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *Proc. ESORICS*, 2010.
[10] X. Cai, X. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proc. ACM CCS*, 2012.
[11] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas, "Circuit fingerprinting attacks: Passive deanonymization of tor hidden services," in *Proc. USENIX Security*, 2015.
[12] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," in *Proc. ACM WPES*, 2013.
[13] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proc. ACM CCS*, 2006.
[14] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-Bayes classifier," in *Proc. ACM CCSW*, 2009.
[15] M. Nasr, A. Bahramali, and A. Houmansadr, "DeepCorr: Strong flow correlation attacks on Tor using deep learning," in *ACM CCS*, 2018.
[16] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proc. ACM WPES*, 2011.
[17] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proc. USENIX Security*, 2014.
[18] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *Proc. NDSS*, 2016.
[19] T. Wang, "High precision open-world website fingerprinting," in *Proc. IEEE S&P*, 2020.
[20] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proc. ACM CCS*, 2018.
[21] S. Bhat, D. Lu, A. H. Kwon, and S. Devadas, "Var-CNN: A data-efficient website fingerprinting attack based on deep learning," in *PETS*, 2019.
[22] C. Wang, J. Dani, X. Li, X. Jia, and B. Wang, "Adaptive fingerprinting: website fingerprinting over few encrypted traffic," in *Proc. ACM CODASPY*, 2021.
[23] W. Cui, T. Chen, and E. Chan-Tin, "More realistic website fingerprinting using deep learning," in *Proc. IEEE ICDCS*, 2020.
[24] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with N-shot learning," in *Proc. ACM CCS*, 2019.
[25] T. Wang and I. Goldberg, "On realistically attacking tor with website fingerprinting," in *Proc. PETS*, 2016.
[26] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail," in *Proc. IEEE S&P*, 2012.
[27] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *Proc. ESORICS*, 2016.
[28] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A congestion sensitive website fingerprinting defense," in *Proc. WPES*, 2014.
[29] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proc. ACM CCS*, 2014.
[30] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *Proc. USENIX Security*, 2017.
[31] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko, "Trafficsliver: Fighting website fingerprinting attacks with traffic splitting," in *Proc. ACM CCS*, 2020.
[32] D. Lu, S. Bhat, A. Kwon, and S. Devadas, "Dynaflow: An efficient website fingerprinting defense based on dynamically-adjusting flows," in *Proc. ACM WPES*, 2018.
[33] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *Proc. USENIX Security*, 2016.
[34] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proc. NDSS*, 2018.
[35] G. Rizothanasis, N. Carlsson, and A. Mahanti, "Identifying user actions from HTTP(S) traffic," in *Proc. IEEE LCN*, 2016.
[36] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "BUFFEST: Predicting buffer conditions and real-time requirements of HTTP(S) adaptive streaming clients," in *Proc. ACM MMSys*, 2017.
[37] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *Proc. USENIX Security*, 2017.
[38] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE TIFS*, 2017.
[39] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing Skype traffic: when randomness plays with you," in *Proc. ACM SIGCOMM*, 2007.
[40] A. Bahramali, R. Soltani, A. Houmansadr, D. Goeckel, and D. Towsley, "Practical traffic analysis attacks on secure messaging applications," in *Proc. NDSS*, 2020.
[41] W. Aqeel, B. Chandrasekaran, A. Feldmann, and B. M. Maggs, "On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement," in *Proc. ACM IMC*, 2020.