Scaling website fingerprinting

1st Vincent Ghiëtte
Cyber threat intelligence lab
Delft University of Technology
Delft, The Netherlands
v.d.h.ghiette@tudelft.nl

2nd Christian Doerr
Hasso Plattner Institute for Digital Engineering
University of Potsdam
Potsdam, Germany
christian.doerr@hpi.de

***Abstract*—Website fingerprinting aims to identify the web page visited by a victim through the analysis of metadata generated by the encrypted flow between web server and victim. A fingerprinting attack can be performed at several locations and scales, ranging from local adversaries such as employers monitoring their employees browsing behavior to state sponsored actors monitoring civilians to uncover their political views. In this paper we show the feasibility of an attacker performing web page fingerprinting at a large scale by introducing a new two-stage fingerprinting method. We evaluate our proposed method using a Wikipedia clone consisting of 828,907 pages, allowing us to show that attackers are not only able to fingerprint pages from different websites but are also able to fingerprint similar pages belonging to the same website. More so, we show that, even though HTTP2 reduces the available metadata compared to HTTP, attackers using our method can achieve an accuracy of 62.21% when fingerprinting pages from our Wikipedia clone. Finally, we show that an attacker can, when taking browsing behavior into consideration, identify victims searching for specific information with an accuracy of 87.4%.**

***Index Terms*—Website fingerprinting, large scale, MinHash, LSH**

I. Introduction

According to a survey conducted by Censys [1] in March 2019 over 91% of the websites in the Alexa top 1 million provide the option to encrypt data using TLS. A common assumption is that the widespread adoption of encryption protects users browsing the Internet against attackers eavesdropping and gathering information about the exchanged information. Unfortunately the opposite is quite true: as shown by [2] [3], attackers can use the metadata generated by the exchanged traffic, such as timing and packet sizes, to identify the page requested by the victim.

The feasibility of an attacker uncovering the content of the exchanged information between user and server is a serious breach of the victim's privacy. An attacker can monitor pages visited by the victim to uncover sensitive information. A person suffering from a health condition might browse the Internet for additional information about his or her condition. An attacker sitting between the victim and server can intercept the traffic and fingerprint which pages were requested, thereby uncovering the medical condition of the victim. Hence, employers or medical insurance companies could use website fingerprinting to monitor their employees / clients to determine their health. At a different scale, oppressing regimes can use website fingerprinting to monitor citizens profiling their political views. Therefore, website fingerprinting can have severe consequences for web users.

Previous work has investigated the effectiveness of website fingerprinting, using metadata extracted from the information exchanged between the victim and web server. Good results are achieved by using the Jaccard similarity of object sizes exchanged between parties. More complex methods have also been employed such a machine learning using packet timings and data flows to identify the requested pages.

Although several methods have been proposed to improve the fingerprinting accuracy, little research has been done to investigate the feasibility of fingerprinting at scale. As mentioned earlier, some attackers, such regimes monitoring civilians, might want to deploy web page fingerprinting at scale. Such attackers would need an efficient fingerprinting method to process the vast amount of generated data. In this work we show that fingerprinting can be scaled up by using an efficient algorithm approximating the Jaccard similarity.

Beside increasing the effectiveness of fingerprinting, most proposed methods in previous works are evaluated using web pages from different websites. Although an attacker can extract information from the identified website, we argue that identifying individual pages from a website provides more information. An attacker will gather more knowledge from identifying which Wikipedia article a victim has visited then knowing the victim has visited the Wikipedia index page. Pages from the same website are likely to share common features such as styling, script files, and objects in general. The overlap in shared features between pages results in similar metadata, hence increasing the difficulty of fingerprinting. We show that, an attacker can still discern which page a victim has visited from a website containing 828,907 pages.

Previous work has shown that pages served using the HTTP protocol can be fingerprinted, however no work has been done investigating the fingerprinting of web pages being served using the HTTP2 protocol. The HTTP2 protocol introduces two new features, with the first feature reducing the number of connections established between the client and server. The second feature reduces the data transmitted between the client and server. Both features reduces the amount of metadata which can be used for fingerprinting. According to research conducted by Zimmermann et al. [4] 5.38M domains have

ISBN 978-3-903176-28-7© 2020 IFIP

served their websites using the HTTP2 protocol indicating that the adoption of the HTTP2 protocol has started. Therefore, in this paper we show that while HTTP2 reduces the amount of metadata fingerprinting web pages is still possible.

The contributions of this paper are as follows:

- We demonstrate that web page fingerprinting can be performed at scale achieving sub linear fingerprinting time, processing 900,000 page visits in 2.18 hours.
- We show that although HTTP2 reduces the available metadata fingerprinting is still possible.
- We demonstrate that fingerprinting pages from the same website is possible with an accuracy of 62.21%.
- We show that an attacker can with 87.4% accuracy, identify victims visiting 3 consecutive pages.

The remainder of the paper is structured as follows: Section II gives an overview of the related work. Section III describes the data collection process and the generation of web page fingerprints. Section IV presents our fingerprinting algorithm. Subsequently, in section V the performance of our algorithm is analyzed after which in section VI we analyze the accuracy an attacker can achieve using our fingerprinting method. Finally, section VII concludes our findings.

II. Related work

In this section we present the related work and briefly summarize their key contributions.

Wagner and Schneider [5] were the first to introduce the concept of encrypted traffic analysis. The authors state that the length of encrypted HTTP GET requests can be used to fingerprint web pages. Using the length of the GET request an attacker can infer the length of the URL of the requested page. Having the length of the URL the attacker can match pages with the same URL length hence identify the page requested by the victim.

Cheng and Avnur [6] continue the analysis of website fingerprinting and introduce the concept of fingerprinting web pages using the page and total object sizes exchanged between the server and client. The authors calculate the HTML page size using the first data stream sent from the server to the client. The total object sizes is calculated using the total traffic sent from the server after the first data stream. The authors fingerprint web pages by matching *(page size, object size)* tuples. Hintz [2] and Sun et al. [3] expand the definition of object sizes introduced by Cheng and Avnur. Both [2] and [3] argue that when a user connects to a web page all files which need to be loaded are sent through separate connections. Therefore, the authors treat each separate connection as a data stream representing an object of a web page. Hintz fingerprints pages by comparing the object sizes and finding the page with the most matching sizes. Sun et al. calculate the Jaccard similarity using the object sizes to find matching pages. By using the Jaccard similarity Sun et al. [3] achieve a fingerprinting accuracy of 75% when matching pages from a data set consisting of 2,191 pages.

Liberatore and Levine [7] also use the Jaccard similarity to match web pages. However, the authors use the sizes and directions of the IPv4 packets instead of using only the object sizes. In addition, the authors fingerprint web page which have been tunneled through an SSH proxy. The authors achieve an accuracy of 88% when using a data set containing 2,000 web pages. Lu et al. [8] also fingerprint web pages sent through a SSH tunnel. Similar to Liberatore and Levine they use the IPv4 packet sizes and directions but add the order in which the packets have been sent. The IPv4 packet sizes, directions and order are represented as a string allowing the authors to use the Levenshtein distance to find matching pages. Testing their method on the same data set as Liberatore and Levin, Lu et al. achieve an accuracy of 81%. Herrmann et al. [9] also analyze the feasibility of website fingerprinting when the victim uses an SSH tunnel. In addition, the authors analyze the influence of using Tor and JonDonym, two privacy enhancing technologies (PET), on website fingerprinting. Herrmann et al. use a multinomial naïve bayes classifier to fingerprint web pages. The authors compare their approach to using the Jaccard similarity for website fingerprinting and achieve an accuracy of respectively 86.66% and 91.63% for a data set consisting of 777 pages loaded through and SSH tunnel.

Panchenko et al. [10] perform website fingerprinting using machine learning. By using support vector machine (SVM) classification they increase the fingerprinting accuracy achieved by [9] on the Tor and JonDonym data sets from respectively 2.95 to 55% and from 20% to 80%. Cai et al. [11] also use support vector machine learning but use Damerau-Levenshtein edit distance as kernel for the SVM achieving similar accuracy to [10]. Wang and Goldberg [12] improve on the research performed by Cai et al. by optimizing the string alignment method. The optimization improves the accuracy rate to 91% on both the data set provided by Cai et al. and their own generated data set consisting of 1000 web pages.

Wang et al. [13] continue to study website fingerprinting on data gathered from browsing to websites using Tor. However, they increase the data set size to 5,000 pages. The authors use a fingerprinting technique based on k-Nearest Neighbor and achieved an accuracy of 85%. Panchenko et al. [14] also increase their data set size and include 111,884 pages. The authors use SVM to perform website fingerprinting and achieve an accuracy of 80%. Hayes et al. [15] achieve similar performance on a data set of comparable size to the one used in [14]. The authors achieve an accuracy of 85% by using random forests as their fingerprinting method.

Previous works have shown that fingerprinting websites using the sizes of the objects exchanged between the server and client is possible and that using the Jaccard similarity as a fingerprinting method can achieve a good accuracy. More so, previous works show that fingerprinting website is possible when different encryption mechanisms are used. Furthermore, recent works have increase the number of pages in the data sets suggesting that fingerprinting at scale is possible.

In the following we continue the analysis of website fingerprinting by evaluating the performance of an two-stage algorithm combining an approximation of the Jaccard similarity and the real Jaccard similarity as our fingerprinting method.

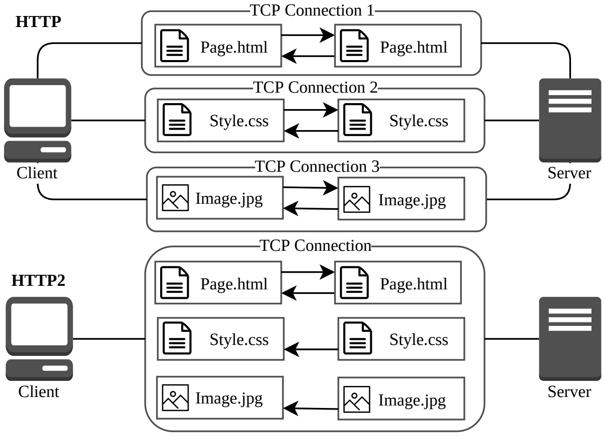

Fig. 1. Key differences between HTTP and HTTP2.

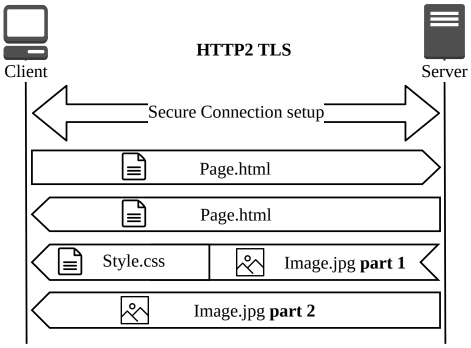

Fig. 2. Data exchange between client and server using HTTP2 and TLS.

We also show that fingerprinting is possible on a larger data set than previously used in research. More so, we evaluate website fingerprinting for pages served using the HTTP2 protocol, which has not been analyzed by previous research.

III. Extracting fingerprinting features

In this paper we show the feasibility of an attacker performing web page fingerprinting on a large set of similar pages belonging to the same website. More so we show that the innovations introduced by the HTTP2 protocol are inadequate to mitigate web page fingerprinting. Therefore, in this section we first analyze the HTTP2 protocol highlighting the innovations which may affect fingerprinting accuracy. Second, we present the features used for fingerprinting web pages. Following we present the methodology used for collecting the required data representing a large number of similar web pages. Finally, we show that pages belonging to the same website are more similar than pages belonging to different websites.

A. *HTTP2 fingerprinting challenges*

According to a survey conducted by Censys [1] 91% of the web pages in the Alexa top million are served using HTTP2. Although related work has shown that fingerprinting pages served using the HTTP protocol is possible, no work has investigated the feasibility of fingerprinting pages which are served using HTTP2. The HTTP2 protocol introduces two new mechanics which influence the data transfer between client and server. Therefore, we analyze the HTTP2 protocol and compare it to HTTP, highlighting the key differences and present their potential impact on fingerprinting.

Figure 1 illustrates the main differences between HTTP and HTTP2. The top part of the figure shows a client loading a web page consisting of a page, a styling sheet and an image using the HTTP protocol. While the bottom part shows the client loading the same web page using the HTTP2 protocol.

The most notable innovation introduced by HTTP2 is the use of a single TCP connection (RFC7540) [16] as shown in figure 1. When loading a page using HTTP, a new TCP connection is established each time the client demands a new resource from the server. New connections are established for exchanging the page.html, style.css and image.jpg files. When using HTTP2, the client reuses the same TCP connection to demand new resources from the server. HTTP2 reduces the number of connection established between the client and server reducing the metadata which can be extracted from the communication between client and server. Fingerprinting web pages is relies on metadata, therefore HTTP2 reduce the amount of information that can be used for fingerprinting.

In addition to using a single connection RFC7540 [16] also specifies the use of a new technique named server push. Server push allows a server to send resources to the client without the client having to request them. The server push technique is implemented to enable a server to send data ahead of time effectively reducing the number of queries sent by the client, and speeding up the loading time of the web page. Figure 1 illustrates a server push, after receiving the request for the page, the server using HTTP2 sends the style sheet and the image without the client requesting it. In contrast, the server using HTTP waits for the client to first establish a new connection and then request the style sheet and image prior to sending the resources. Again, the HTTP2 protocol reduces the amount of metadata which can be extracted from the exchange between client and server. By removing the necessity for a client to request data, the number of transaction made by the client is reduced, in this case from three to one request. HTTP2 reduces the amount of exchanged data and thus the amount of metadata which can be used for fingerprinting.

Previous work studying HTTP has used data flows to fingerprint websites, however in HTTP2 the data is effectively tunneled through a single connection removing the possibility to deduce different flows. More so the introduction of the server push mechanism reduces the exchanged data and can potentially alter the order of in which the data is sent as the server may push resources at any given time effectively reducing the amount of metadata. These two properties of HTTP2 should increase the difficulty of website fingerprinting, however we show in the following that fingerprinting is still possible.

B. Fingerprinting features

In the previous section we have shown that HTTP2 reduces the available metadata which can be used to fingerprint web pages. Therefore, we have opted to base our fingerprinting method on the sizes of the objects exchanged between client and server. The transferred object sizes are not be affected by the server push, nor by the tunneling performed by HTTP2.

HTTP2 can be used in combination with TLS to encrypt the traffic. More so common browsers such as Firefox and Chrome have expressed their intent to only allow the encrypted use of HTTP2. Therefore, we investigate how the objects are encrypted using HTTP2 in combination with TLS. Figure 2 shows a client requesting a web page from a server using HTTP2 and TLS. First a secure TLS connection is established, after which the client sends a packet containing a TLS record consisting of the encrypted request for a web page. The server responds by sending a packet containing a TLS record with the encrypted page data. Following, the server pushes additional resources to the client by sending two more packets containing records for the style sheet and image.

We are interested in the record sizes exchange between the server and client as they offer a representation of the exchanged object sizes. Extracting the records sizes can be achieved by inspecting the record header which contains a field with the record length (RFC5246) [17]. Even though extracting the record size is a simple task, some precaution is required. Figure 2 illustrates the segmentation of records across different packets, with the record containing the image being split across two packets. Ignoring record segmentation can lead to processing the seconds part of the image as a new record providing a fictive record size. Therefore, prior to processing the records we reconstruct the IPv4 packet stream reassembling the segmented records eliminating the potential of false header information.

Using the record sizes we can infer the sizes of the objects exchanged between the server and client. However not all information exchanged between client and server are objects. During a TLS connection management information is exchanged between server and client. These management records do not represent exchanged objects, and should therefore be ignored when fingerprinting. Management records, and other record types not related to the actual objects exchanged, are filtered out by processing the TLS header and consulting the type field. All non data type records are removed from the collected data resulting in record sizes representing the exchanged objects between client and server. In section IV we present our algorithm which uses the record sizes sent by the client and the server separately to fingerprint web pages.

C. Data collection

In order to test the scalability of our proposed fingerprinting method we require a data set containing a large amount of web pages. Further, the pages in the data set must originate from the same website as we show that an attacker is capable of identifying pages from the same website. Finally, the pages should be served using HTTP2 as we show that an attacker can fingerprint pages being served using the aforementioned protocol.

To ensure we have a large number of pages belonging to the same website we selected to use Wikipedia as a base for our data set. Furthermore, we hosted a mirror of Wikipedia in order to circumvent scraping limits imposed by the Wikimedia Foundation managing the Wikipedia websites. Besides circumventing scraping limits, a self hosted solution offers the ability to select the HTTP2 protocol for serving web pages.

Textual dumps of different languages of Wikipedia websites are offered by Wikimedia [18]. While the dumps of the corresponding media files are available on the Internet Archive [19]. Searching for a matching textual dump and media dump resulted in us using a Spanish version of Wikipedia with the textual dump and media dump dating from respectively 2011 and 2012. The time difference between both dumps caused some articles, which we excluded, to have missing media files resulting in a data set containing 828,907 articles.

With the website selected, we need to generate a data set enabling us to test the feasibility of website fingerprinting at scale. An attacker performing website fingerprinting will record the data transferred between a victim loading a web page and the server serving that page and store it into a trace, with a trace being a raw representation of captured data. Following, the attacker extracts metadata from the trace and finds the closest match in a database containing traces of earlier loaded pages. To generate the traces of the victims visiting the web pages, and to generate a trace database of the attacker we need to crawl the entire Wikipedia website generating the traffic for the traces. In order to crawl all the articles on the Wikipedia clone, we extract a list of pages using Wikicrush [20] allowing us to visit the articles using a browser. We automate the browsing process by feeding the article URLs to Selenium [21] which calls the Chrome driver [22] in incognito mode. The traces are generated using the tcpdump [23] capturing the raw data. For each page we generated three traces, one representing the victims browsing to the page and two for the database of the attacker. Having captured 2,486,721 traces representing the 828,907 pages from our data, we proceed by extracting the record sizes for each trace representing the objects exchanged between server and client.

D. Web page similarity

Related work has shown that fingerprinting web pages is possible, however all work achieved high accuracy using pages from different websites or only a couple of pages from the same website. We claim that although pages from the same website are more similar fingerprinting is still possible.

Pages originating from the same website are more likely to have similar fingerprints as pages from the same site are likely to share resources. These shared resource can be logos, style sheets, scripts and other content. As fingerprinting is based on the metadata of the exchanged information between client and server these shared resources will cause web pages sharing

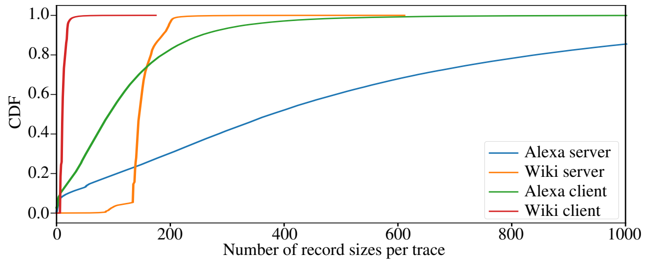

Fig. 3. The cumulative distribution function of number of record sizes for the traces in the Wikipedia and Alexa top 100,000 data sets. A distinction is made between number of records sent by the server and sent by the client.

those resources to have similar fingerprints. For instance a shared script file will not change in size depending on the loaded page hence result in the same record size leading to a similar page fingerprint.

In order to investigate the similarity of pages originating from the same website compared to those originating from different websites we need a data set of pages from different websites. Using the same methodology as for creating the Wikipedia data set, we created a data set for the Alexa top 100,000. After sanitizing the data set and removing web pages not loaded using the HTTP2 protocol, the data set contains 55,212 pages with three traces per page.

Having generated both data sets, we now need to establish comparison metrics to measure the similarity of pages belonging to the same set. Our fingerprinting method relies on the sizes of the records sent by the server and those sent by the client. The record sizes provide an indication of the exchanged object sizes which is used to determine page similarity. If pages are similar, they will most likely share objects, thus having the same record sizes in their fingerprints. Analogue to the diversity in record sizes, the similarity of pages can also be measured in terms of objects loaded per page. Similar pages will have the same amount of objects, therefore the number of records in a page traces is also an indication of similarity. We will use these two metrics to analyze the similarity between pages in the two data sets.

Figure 3 shows the cumulative distribution function (CDF) of the number of records contained in each trace for both the Alexa and Wikipedia data sets. A distinction is made between the number of records sent by the client and server as our proposed algorithm uses both separately to fingerprint web pages. A steep incline in the CDF curves of both Wikipedia client and server records shows that the majority of traces have the same amount of records. On average the client sends 16 records while the server sends 159 records with respective standard deviations of 5.6 and 24.7. The steep incline in the CDF curve shows that most of the trace contains a similar amount of records indicating that pages are similar in terms of objects. The contrary is true for the pages in the Alexa data set, where the CDF plot shows that for both the number of records sent by the server and by the client have more variation. On average the server sending 451 records and the client sending 121 records with respective standard deviations of 685.36 and 118.39.

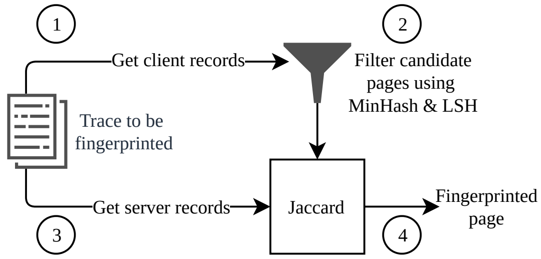
1
2
Get client records
Filter candidate pages using MinHash & LSH
Trace to be fingerprinted
Get server records
Jaccard
Fingerprinted page
3
4

Fig. 4. Proposed fingerprinting method

Inspecting the number of records exchanged in both data sets shows that pages from Wikipedia are more similar than pages from the Alexa top 100,000 in terms of exchanged records. The similarity of pages in the Wikipedia data set is further strengthened by the number of unique record sizes in both data sets. In the Wikipedia data set we observe 990 and 18,690 distinct record sizes exchange by respectively the client and server. For the Alexa data set we observe 7,736 and 28,987 distinct record sizes exchanged by respectively the client and server. The larger number of unique record sizes in the Alexa data set is an indication that the object sent between client and server are more varied than in the Wikipedia data set.

Both in terms of number of records and number of unique record sizes the pages loaded from the Wikipedia website exhibit less variation than the ones loaded from the Alexa top 100,000. This lack of variation confirms that pages loaded from Wikipedia have more similar fingerprints than the pages in the Alexa data set. Although pages originating from Wikipedia are more similar in terms of objects sent and the sizes of sent object, we show in section V that fingerprinting is still possible.

IV. Fingerprinting Method

In this section we introduce our fingerprinting method. First, we analyze the Jaccard similarity, a method used in several related works achieving 88% accuracy, and show that it does not scale. Second, we propose a two-stage algorithm using using MinHash and locality sensitive hashing in combination with the Jaccard similarity improving scalability. Third, we analyze the influence of varying the settings required by MinHash and locality sensitive hashing on the fingerprinting results and space requirements.

A. Established fingerprinting method

Website fingerprinting is performed by comparing metadata extracted from a trace and finding the closest matching trace in a database containing other traces. Related work has shown that using the Jaccard similarity to fingerprint web pages is effective, with Sun et al. [3] achieving an accuracy rate of 75% and later Liberatore and Levine [7] achieving an accuracy rate of 88%. The Jaccard similarity between two pages is calculated using the record sizes sent by the client and those sent by the server. The number of common record sizes between both

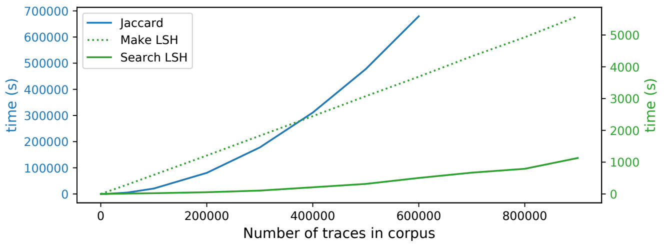

Fig. 5. Time comparison of fingerprinting using Jaccard index, and MinHash and LSH combination.

pages is divided by the number of unique record sizes shared by both pages, resulting in a number between [0, 1] indicating the similarity between both pages. Although intuitive, using the Jaccard similarity to identify pages is computationally expensive. Each time a trace needs to be fingerprinted the Jaccard similarity needs to be calculated for each trace in the data set. Hence increasing the number of traces to fingerprint and the number of traces in the data set results in a quadratic time increase.

In order to investigate whether the method scales on larger data sets we implement the Jaccard similarity calculation for the traces in C++ using the standard library's library implementation of the multi set and its intersection and union methods. Figure 5 shows the computing time in seconds of running full comparison for the client record of a number of traces in a data set, meaning all traces in the data set are compared to each other using the client records in order to find candidate matches. Computing all Jaccard similarities between the traces in a data set of 900,000 traces took 7.8 days to complete. This long processing time translates into an attacker monitoring a million victims at the same time visiting one page would finish processing the data after one week.

Clearly using the Jaccard similarity for website fingerprinting is not feasible when dealing with large amount of traces. Therefore, in the following subsection, we propose an alternative which allows for large scale web page fingerprinting.

B. Trading off look-up speed with space

In the previous subsection we have shown that using the Jaccard similarity as proposed in related work is not feasible for fingerprinting at scale. Therefore, we propose an alternative eliminating the necessity to compare all traces in the database to find the closest matching trace, trading in the quadratic time complexity for space complexity.

Using MinHash [24], an approximation algorithm for the Jaccard similarity, allows us to use locality sensitive hashing (LSH) [25] to look up similar traces. LSH is designed to find similar sets of items in a large data set achieving sub linear matching times. Therefore, combining MinHash and LSH should result in a faster web page fingerprinting as not all traces in the data set need to be compared.

In order to test the scalability of the MinHash and LSH combination, a fingerprinting algorithm is built using the datasketch library [26] following the example implementation for MinHash and LSH. Figure 5 shows the computing time in seconds of running full comparison for the client records of a number of traces in a data set. The performance is depicted by two lines, the dotted line indicates the time taken to build the lookup database which should be performed only once by the adversary prior to the attack. The solid line indicates the lookup time that would be performed in real time during the attack. Making the LSH database is linear in time, and takes 1.5 hours to complete for a data set containing 900,000 traces. More important is the lookup time as it dictates how fast a trace can be fingerprinted. Figure 5 shows that the lookup times are sub linear, fingerprinting 900,000 traces in a data set of 900,000 traces only took 18.8 minutes.

Although using MinHash and LSH we achieve sub linear fingerprinting times using the client records, using MinHash for fingerprinting web pages using the server records is not possible. The LSH database for the client records took 20GB of space while the LSH database for the server records exceeded 148GB, exhausting the available memory on our server. The MinHash and LSH algorithm make a trade off between speed and memory consumption. For each trace, MinHash requires to store a number of vectors of the length of the total number of unique record sizes in memory. As shown in previous section the number of unique server record sizes is 18690 compared to 990 unique client record sizes. Therefore, the MinHash algorithm is not suitable for fingerprinting traces using the server records as it grows with the number of unique record sizes.

We therefore use a two-stage approach to fingerprint traces as depicted in figure 4. First the candidate pages are filtered using the MinHash and LSH approach using the client records. Then the server records of the trace are compared using the Jaccard similarity with the traces of the candidate pages. The page or pages with the highest Jaccard similarity is then selected and proposed as the identified page. Applying our two-stage approach resulted in fingerprinting a single trace in 0.00872 seconds, and processing the entire 900,000 traces took 2.18 hours.

C. Accuracy vs. space requirements

Using MinHash and LSH algorithms requires setting two parameters. Namely, the number of permutations used by MinHash and the Jaccard similarity threshold used by LSH. In order to measure the influence of the parameters we created a subset of 30,000 traces and ran our fingerprinting algorithm with different parameters.

Table I shows the fingerprinting accuracy and space requirements for different permutation and threshold values. The table shows the sizes of the LSH database, the percentage of correctly and incorrectly identified pages. The table also shows the percentage of pages for which no definite match was found, and the average number of candidate pages.

First, we analyze the effect of changing the number of permutation used by the MinHash algorithm as it should affect the space requirement. We select 4 values 64, 128, 256, and 512, and set the minimum Jaccard similarity threshold to 0.8

TABLE I
RESULTS FOR VARYING THE THRESHOLD VALUE AND NUMBER OF PERMUTATIONS USED IN THE MINHASH AND LSH ALGORITHM.

Threshold	0.8				0.6	0.7	0.8	0.9
Permutations	64	128	256	512	512			
LSH database size (MB)	29	59	127	248	259	253	248	245
Correct (%)	29.35	27.93	28.35	30.86	8.10	16.70	30.86	48.67
Inconclusive (%)	70.49	71.90	71.51	68.99	91.82	83.21	68.99	51.10
Candidate pages	26.03	29.62	26.01	28.04	42.21	37.13	28.04	30.05
Incorrect (%)	0.14	0.15	0.13	0.14	0.06	0.07	0.14	0.22

for the LSH algorithm. From the value in table I we can see a clear linear increase in terms of space required for the LSH database, while the accuracy of the algorithm remains stable.

Second, we analyze the effect of changing the threshold value as it should affect the accuracy of the algorithm. We select 4 threshold values 0.6, 0.7, 0.8, 0.9 and fix the number of permutation to 512. From the value in table I we can see a clear increase in the percentage of correctly fingerprinted pages when the threshold value increases. The overall trend is that increasing the threshold value results in higher accuracy for the algorithm. More so, increasing the threshold value also reduces the number of candidate pages which increases the speed of our two-stage fingerprinting algorithm.

After analyzing the different parameters values we decided to use 512 as number of permutation and 0.8 as a threshold value. We use 512 as number of permutations as in theory it should perform better when the number of pages in the data set increases, and the LSH database for all 828,907 pages is only 20GB and fits into memory. Furthermore, we select 0.8 as threshold value as it reduces the number of candidate pages while still offering good accuracy.

V. FINGERPRINTING RESULTS

In this section we present the results of fingerprinting all 2,486,721 traces in our data set showing the accuracy of our algorithm on a large set of similar web pages. For each trace in the data set we used our two-stage fingerprinting method to determine the corresponding page of the trace. Our proposed method first matches the corresponding candidate pages of the fingerprinted trace using the client record sizes. In the second step the server record sizes of the trace are compared to the server record sizes of the candidate pages using the Jaccard similarity. Pages with the highest Jaccard similarity are then considered to be the ones corresponding to the trace.

Figure 7 shows the results of the fingerprinting process in the form of a Sankey diagram. The diagram is split up into three steps: *Start*, *MinHash & LSH*, and *Jaccard*, each step representing a phase in the fingerprinting process. The *Start* step represents the beginning of the fingerprinting, in this step all the traces are not identified as a particular page. The *MinHash LSH* step shows the results after fingerprinting the traces using the MinHash LSH approach. After fingerprinting, we see that 7.25% of the traces are correctly matched to their corresponding page meaning the MinHash and LSH stage of the algorithm has reduced the number of candidate pages to one. However, for 1.16% of the trace the MinHash LSH algorithm did not provide any output, and 4.46% of the traces are identified as the wrong page. Leaving 87.13% of the traces being matched with the correct page and one or more incorrect ones. Although the number of traces with an inconclusive result is high after the first stage of the fingerprinting method, the number of candidate pages is significantly reduced. On average the Minhash and LSH algorithm reduced the number of candidate pages from 828,907 to 2,885 pages reducing the computing time in the second stage of the algorithm.

After having fingerprinting the traces using the MinHash and LSH approach we use the Jaccard similarity to fingerprint traces for which no conclusive result is provided. The results of fingerprinting the remaining traces using the Jaccard similarity is shown in the *Jaccard* step of figure 7. After fingerprinting 21.82% of all the traces is identified as the wrong pages, and 15.97% of the traces have been matched with both a correct and one or more incorrect pages. However 62.21% of all traces have been matched with the correct page.

Applying our two-stage fingerprinting method to a data set consisting of 828,907 pages we are able to fingerprint pages with a probability of 62.21%. In the following section we show that an attacker leveraging our method is able to monitor victims browsing the web.

VI. INCORPORATING BROWSING BEHAVIOR

In this section we show the feasibility of an attacker fingerprinting at scale by evaluating the accuracy achieved by an attacker applying our proposed fingerprinting method. Two attack scenarios are taken into consideration, one in which the attacker monitors victims browsing the web randomly visiting pages. The second scenario evaluates the attacker monitoring victims consecutively visiting forbidden pages.

A. Attacker monitoring browsing behavior

An attacker deploying website fingerprinting is interested in identifying victims with a certain preference for a topic. In the case of an oppressing regime, the attacker is interested in identifying civilians which have deviating political views. To do so the regime will compose a fingerprinting database consisting of pages related to unwanted political views. Then the regime will start collecting the traces of civilians visiting web pages and fingerprint them using the composed database. When one or more traces of a civilian matches a fingerprint in the database the regime knows that the civilian shows interest

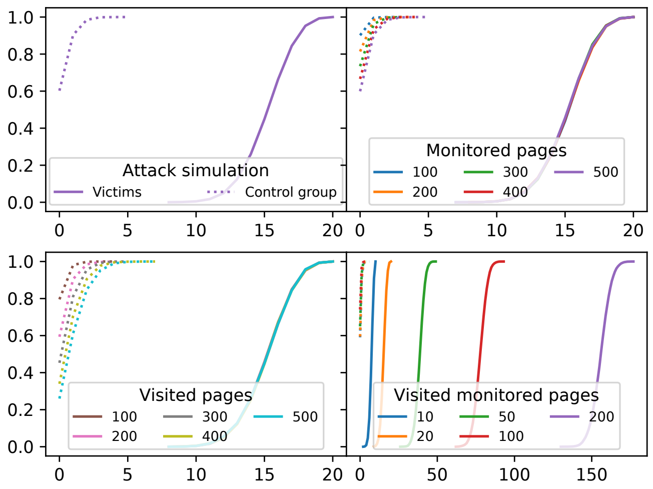

Fig. 6. Accuracy of fingerprinting attacks under different circumstances.

in a political view and can, by the number of matches, estimate the level interest.

To evaluate the success rate of an attacker we simulate a fingerprinting attack. In this attack we simulate 10,000 victims visiting a 200 random pages from our entire data set. Randomizing the selected pages ensures that the simulation is a general representation of an attack with the victims visiting a varied collection of pages. From those 200 visited pages, 20 belong to the database of the attacker, with the attacker having constructed a database of 500 web pages.

The solid line in top left plot of figure 6 shows the results of the simulation. The graph represents the CDF of the number of pages correctly fingerprinted. Simulation results reveal that all victims are identified by the attacker as having visited at least 8 monitored pages, with 74.12% of the victims being identified as having visited at least 15 pages. The simulation shows that an attacker can identify all simulated victims using a threshold of 8 identified pages. However, the question arises how many people would falsely be classified as having visited monitored pages when in fact they did not. In order to investigate the number of falsely flagged victims we run the same analysis, but this time with the victims not visiting monitored pages effectively forming a control group. The dashed line in the top left plot of figure 6 shows the CDF of incorrectly identified pages for the control group. The simulation of the control group results in the attacker identifying all the victims as having visited 6 or less monitored pages with 99.33% of the victims as having visited 3 or less monitored page.

Our simulation shows that an attacker can identify all victims having visited 20 monitored pages without any false positives. However increasing the number of pages visited by the victims, or number of monitored pages by the attacker might influence the accuracy of the attack. Therefore, we have simulated 15 more attacks with different values for the number of pages visited, pages monitored and pages visited monitored. Figure 6 shows the results of the simulations. For each plot the dashed line represents the CDF of the number of incorrectly identified pages (control group), and the solid line representing the CDF of the number of correctly identified pages.

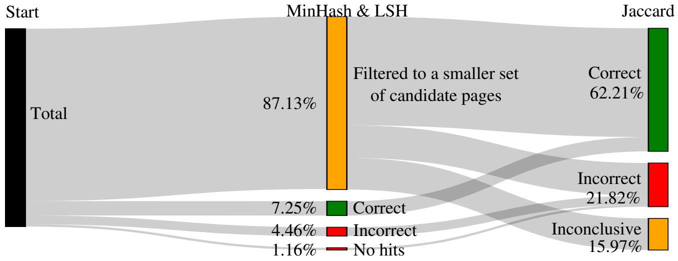

Fig. 7. Fingerprinting accuracy throughout the different phases of our algorithm.

The top right plot shows simulation results where the number of monitored pages in the attacker's database varies. It is clear that varying the number of monitored pages does not influence the accuracy of the attack. Similarly, varying the number of visited pages does not influence the accuracy as shown by the bottom left plot. The bottom right plot shows that increasing the number of visited monitored pages does increase the number of correctly identified pages.

The results of the different simulation show that increasing the variables in a attack simulation do not affect the accuracy of the fingerprinting. As a result an attacker can, using correct thresholds, use our proposed approach to fingerprint at scale. In the following section we show that an attacker, assuming victims visit monitored pages successively, can avoid setting a threshold.

B. Attacker monitoring victims following links

In the previous section we have shown that an attacker can identify victims visiting monitored pages. However, the attacker needs to set a threshold in order to avoid falsely identifying victims. Typically, a person would not visit a page out of nothing, but follow a set of links towards pages of the same topic. An attacker can use this behavior to his advantage, instead of counting the number of monitored pages visited by the victims, the attacker can count the number of consecutive pages a victim has visited. To analyze the proposed attack we simulate an attacker monitoring victims consecutively visiting between 2 or 5 pages as well a control group which did not visit any monitored page.

Table II shows the result of 10,000 victims visiting between two and five monitored pages, and the results for a control group of 10,000 victims having visited 200 non monitored pages. The table shows the percentage of victims being identified by the attacker as having visited a number of pages consecutively. From the results becomes clear that victims visiting more consecutive pages are more likely to be identified as doing so by the attacker. For instance, the attacker is able to identify 87,4% (39.1% + 48.3%) of the victims having visited three pages as having visited two or more pages. Whereas the attacker is able to identify 96,4% (18.8% + 41.5% + 36.1%)

TABLE II
PERCENTAGE OF VICTIMS IDENTIFIED AS HAVING VISITED A NUMBER OF CONSECUTIVE PAGES.

	Pages identified by the attacker					
	0	1	2	3	4	5
Control group	57.2	30.8	9.8	2.1	0.1	-
Visited 2 pages	4.7	33.3	62.0	-	-	-
Visited 3 pages	1.1	11.5	39.1	48.3	-	-
Visited 4 pages	0.3	3.3	18.8	41.5	36.1	-
Visited 5 pages	-	0.8	7.0	21.9	39.6	30.7

of the victims having visited four pages as having visited two or more pages.

The results from the control group show that an attacker will falsely identify 12.0% (9.8% + 2.1% + 0.1%) of people not having visited monitored pages as having visited 2 or more. Given the results from the simulation we can say that an attacker is able to identify victims visiting 3 or more pages with an detection rate of 87.4% and a false positive rate of 12%. An attacker willing to increase the true positive rate or false positive rate could change the number of consecutively detected pages.

VII. CONCLUSION

In this paper we have presented a novel approach for website fingerprinting based on the Jaccard similarity and an approximation of the Jaccard similarity. Using our approach on a data sets of 828,907 pages we are able to achieve a fingerprinting accuracy of 62.21%. Furthermore we have show that the new mechanics introduced by the HTTP2 protocol does not make fingerprinting web pages impossible.

Finally we show the feasibility of an attacker performing web page fingerprinting at scale. We show that an attacker is able to identify victims with an accuracy of 87.4%.

REFERENCES

[1] Censys, "Internet-wide scan data repository," 2019. [Online]. Available: https://censys.io/

[2] A. Hintz, "Fingerprinting websites using traffic analysis," in *Privacy Enhancing Technologies*, R. Dingledine and P. Syverson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 171–178.

[3] Qixiang Sun, D. R. Simon, Yi-Min Wang, W. Russell, V. N. Padmanabhan, and Lili Qiu, "Statistical identification of encrypted web browsing traffic," in *Proceedings 2002 IEEE Symposium on Security and Privacy*, May 2002, pp. 19–30.

[4] T. Zimmermann, J. Rüth, B. Wolters, and O. Hohlfeld, "How http/2 pushes the web: An empirical study of http/2 server push," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, June 2017, pp. 1–9.

[5] D. Wagner and B. Schneier, "Analysis of the ssl 3.0 protocol," in *Proceedings of the 2Nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, ser. WOEC'96. Berkeley, CA, USA: USENIX Association, 1996, pp. 4–4.

[6] H. Cheng and R. Avnur, "Traffic analysis of ssl encrypted web browsing," 1998.

[7] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 255–263.

[8] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *Computer Security – ESORICS 2010*, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 199–214.

[9] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 31–42.

[10] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, ser. WPES '11. New York, NY, USA: ACM, 2011, pp. 103–114.

[11] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 605–616.

[12] T. Wang and I. Goldberg, "Improved website fingerprinting on tor," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, ser. WPES '13. New York, NY, USA: ACM, 2013, pp. 201–212.

[13] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, ser. SEC'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 143–157.

[14] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[15] J. Hayes and G. Danezis, "K-fingerprinting: A robust scalable website fingerprinting technique," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16. USA: USENIX Association, 2016, p. 1187–1203.

[16] M. Belshe, M. Thomson, and R. Peon, "Hypertext transfer protocol version 2 (http/2)," 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7540

[17] T. Dierks and E. Rescorla, "Rfc 5246-the transport layer security (tls) protocol version 1.2," Internet Requests for Comments, RFC Editor, RFC 5246, August 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5246.txt

[18] Wikimedia Foundation, "Wikimedia dumps." [Online]. Available: https://dumps.wikimedia.org

[19] "Internet archive." [Online]. Available: https://archive.org

[20] H. Tristan, "Wikicrush," 2016. [Online]. Available: https://github.com/trishume/wikicrush

[21] Selenium HQ, "Selenium," 2018. [Online]. Available: https://github.com/SeleniumHQ/selenium

[22] Chromium project, "Chrome driver," 2018. [Online]. Available: http://chromedriver.chromium.org

[23] The Tcpdump Group, "tcpdump," 2018. [Online]. Available: https://github.com/the-tcpdump-group/tcpdump

[24] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *Journal of Computer and System Sciences*, vol. 60, no. 3, pp. 630 – 659, 2000.

[25] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, vol. 51, no. 1, pp. 117–122, Jan. 2008.

[26] E. Zhu, "datasketch: Big data looks small," 2019. [Online]. Available: https://ekzhu.github.io/datasketch/