International Journal of Humanities and Social Science Invention
ISSN (Online): 2319 – 7722, ISSN (Print): 2319 – 7714
www.ijhssi.org ||Volume 6 Issue 3|| March. 2017 || PP.70-87

A Timing Information Based Attack on Web Traffic Analysis

Sriraksha T.A[1], Noor U saba[2], Shivaraj kumar T.H[3]
[1]*MTech, Dept. of CS&E, C.Byregowda Institute Of Engineering Kolar.*
[2]*MTech, Dept. of CS&E, C.Byregowda Institute Of Engineering Kolar.*
[3]*MTech, Dept. of CS&E, C.Byregowda Institute Of Engineering Kolar.*

ABSTRACT: *We introduce an attack against encrypted web traffic that makes use solely of packet timing information on the uplink. This attack is thus imperviable to existing packet padding defenses. Additionally, not like existing approaches, this timing-only attack doesn't need the knowledge of the start/end of web fetches then is effective against traffic streams. We tend to demonstrate the effectiveness of the attack against both wired and wireless traffic, achieving mean success rates in more than 90%. Additionally to being of interest in its own right, this timing-only attack serves to spotlight deficiencies in existing defenses then to areas wherever it might be useful for virtual private network (VPN) designers to focus more attention.*

Keywords: *Timing-only attacks, Network privacy, traffic analysis, website fingerprinting.*

I. INTRODUCTION

In this paper we have a tendency to take into account an attacker of the kind illustrated in Figure 1. The attacker will notice the time once packets traverse the encrypted tunnel within the transmission direction, however has no different info concerning the clients' activity. The attacker's objective is to use this information to guess, with high likelihood of success, the online sites that the client visits. What's distinctive concerning the attack thought about here is that the attacker depends entirely on packet timestamp info whereas the antecedently reported attacks against encrypted internet traffic have primarily created use of observations of packet size and/or packet count info.

Our interest in timing-only attacks is twofold. Firstly, packet padding may be a comparatively easy defence against attacks that rely primarily on packet size, and so is presently either already out there or being enforced during a variety of popular virtual private networks (VPN) [2]. Secondly, different attacks supported packet counting [2], [3] are insensitive to packet artefact defences however need partitioning of a packet stream into individual web fetches so as for the quantity of packets related to every web fetch to be determined, which can be extremely difficult in observe on links where there aren't any clear pauses between web fetches.

In distinction, packet timing-based attacks aren't solely for the most part unaffected by packet padding defences however conjointly, as we'll show, don't need partitioning of the packet stream. Hence, they're probably a much necessary category of attack against current and future VPNs. whereas some work has been administrated using inter-arrival time information to classify the application (HTTP, IMAP etc.) [8], to our data, there's no previous work coverage use of timing information alone to construct a prosperous attack against encrypted web traffic.

The main contributions of this paper are as follows: (i) we have a tendency to describe an attack against encrypted web traffic that uses packet timing information alone, (ii) we tend to demonstrate that this attack is extremely effective against each wired and wireless traffic, achieving mean success rates in far more than 90th over Ethernet and wireless tunnels and successful rate of 58 against Tor traffic, (iii) we have a tendency to additionally demonstrate that the attack is effective against traffic streams i.e. back to back web page fetches wherever the packet boundaries between fetches are unknown.

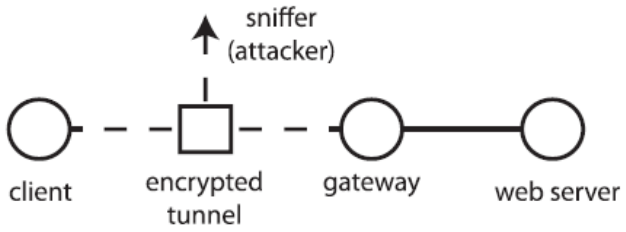

Fig. 1. Schematic illustrating attacker of the type considered. A client machine is connected to an external network via an encrypted tunnel (ssh, SSL, IPSec *etc*.). The attacker can detect the time when packets traverse the tunnel in the uplink direction, but has no other information about the clients activity.

In addition to being of interest in its claim, notably in sight of the powerful nature of the attack, this timing-only attack additionally serves to focus on deficiencies in existing defences and then to areas wherever it'd be helpful for VPN designers to focus more attention. we have a tendency to note that, complementary to this work, in [3] it's demonstrated that once web fetch boundaries among a packet stream are noted then an NGRAM approach using packet count along with uplink/downlink direction data is additionally sufficient to construct a good attack against encrypted web traffic despite packet padding.

Hence, we can conclude that (i) uplink/downlink packet ordering and web fetch boundaries and (ii) uplink/downlink packet timing data are each sensitive quantities that need to be protected by a secure encrypted tunnel. Packet padding doesn't defend these quantities. Guiding defences against these two sets of packet stream options so looks a very important direction for future work.

II. RELATED WORK

The general topic of traffic analysis has been the topic of much interest, and a large body of literature exists. a number of the earliest work specifically focused on attacks and defences for encrypted web traffic seems to be that of Hintz [7], that considers the Safe Web encrypting proxy. During this setup (i) web page fetches occur sequentially with the beginning and finish of every web page fetch known, and for every packet (ii) the client-side port number, (iii) the direction (incoming/outgoing) and (iv) The scale is determined. A web page signature is built consisting of the combination bytes received on every port (calculated by summing packet sizes), effectively such as the amount and size of every object within the web page. In [15] it's equally assumed that the number and size of the objects in a very web page may be determined and using this information a classification success rate of 75th is reported. Subsequently, Bissias et al. [1] thought of an encrypted tunnel setup where (i) web page fetches occur sequentially with the beginning and finish of every online page fetch better-known, and for every packet (ii) the size, (iii) the direction (incoming/outgoing) and (iv) the time (and thus additionally the packet ordering) is determined. The sequence of packet inter-arrival times and packet sizes from a web page fetch is employed to form a profile for every online page in a target set and also the cross correlation between an determined traffic sequence and also the stored profiles is then used as a measure of similarity. A classification accuracy of 23rd is determined once employing a set of 100 web pages, rising to 400th once restricted to a smaller set of web pages. Most later work has adopted primarily identical model as [1], creating use of packet direction and size information and assumptive that the packet stream has already been divided into individual web page fetches. as an example in [16] the timing information isn't thought of within the feature set, thus the attack is countered with defences like BuFLO in [3] resulting in successful rate of solely 100 percent. In [6] and [10] Bayes classifiers based on the direction and size of packets are considered whereas in [14] an SVM classifier is proposed. In [11] classification supported direction and size of packets is studied using Levenshtein distance as the similarity metric, in [13] using a Gaussian Bag-of-Words approach and in [16] using K - NN classification. In [2] using a SVM approach a classification accuracy of over 80th is reported for each SSH and Tor traffic and also the defences thought of were typically found to be ineffective. Similarly, [3] considers Bayes and SVM classifiers and finds that a range of proposed defences are ineffective. In [5] remote inference of packet sizes from queueing delay is studied.

III. ANATOMY OF A WEB PAGE FETCH

When traffic is carried over an encrypted tunnel, like a VPN, the packet source and destination addresses and ports and therefore the packet payload are hidden. we tend to also assume here that the tunnel pads the packets to be of equal size, in order that packet size information is additionally concealed, and that the beginning and end of an individual web fetch may additionally be concealed e.g. once the web fetch is embedded in a larger traffic stream.

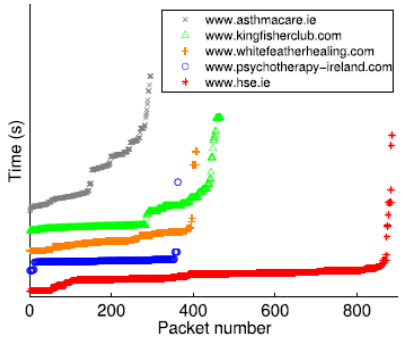

Fig. 2. Time traces of uplink traffic from 5 different Irish health-related web sites are shown. It can be seen that the web site time traces exhibit distinct patterns. The traces are shifted vertically to avoid overlap and facilitate comparison.

An assailant sniffing traffic on the encrypted tunnel is thus ready solely to look at the direction and timing of packets through the tunnel, i.e. to observe a sequence of pairs , k = 1, 2, • • • in the uplink or downlink direction. Our experiments on use of uplink, downlink and uplink + downlink traffic recommend that downlink traffic provides no further information concerning timing patterns over uplink traffic. The reason is that the timing of ACKs in uplink traffic is correlated to it of downlink packets which implies that using solely uplink traffic provides sufficient information. Furthermore it may be easier for an eavesdropper to access unmodified uplink traffic on the first hop, (given the traffic comes immediately from the source, whereas the corresponding downlink traffic may well be morphed using inter-flow transformations e.g. flow mixing, split and merge [17]). We tend to therefore focus on an attacker which will solely observe the timestamps , k ∈ Kup := : dκ = −1} related to uplink traffic

Figure 2 plots the timestamps of the uplink packets sent throughout the course of winning five completely different health-related web content (see below for details of the measurement setup). The x –axis indicates the packet number k within the stream and therefore the y-axis the corresponding timestamp tk in seconds. It may be seen that these timestamp traces are clearly completely different for every web site, and it's this observation that motivates interest in whether or not timing analysis might by itself (without extra information like packet size, uplink/downlink packet ordering etc.) be sufficient to successfully de-anomymise encrypted web traffic.

To gain insight into the variations between the packet timestamp sequences in Figure 2 and, significantly, whether or not they are genuinely associated with characteristics of every web page rather than to other factors, it's useful to think about the method of fetching a web page in more detail. To fetch a web page the client browser starts by opening a TCP connection with the server indicated by the URL and problems an hypertext transfer protocol GET or POST request to that the server then replies. As the client parses the server response it problems additional GET/POST requests to fetch embedded objects (images, css, scripts etc.).

These additional requests may be to different servers from the original request (e.g. once the object to be fetched is an advert or is hosted in a separate content-delivery network), in which case the client opens a TCP connection to every new server in order to issue the requests. fetching of these objects might successively trigger the fetching of further objects. Note that asynchronous winning of dynamic content mistreatment, e.g. AJAX, will result in a fancy sequence of server requests and responses even after the page has been rendered by the browser. Also, generally the TCP connections to the various servers are held open till the page is totally loaded so that they can be reused for later requests (request pipelining during this method is nearly universally employed by trendy browsers).

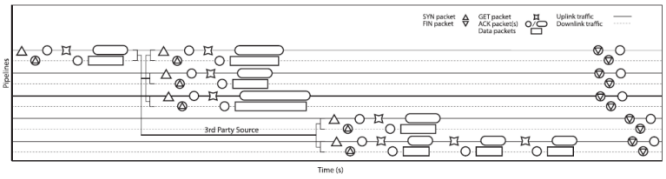

Fig. 3. This figure represents a typical web site query. It starts by requesting the index page. Then as the browser parses through this page more objects are fetched in parallel. Some objects may also be outsourced to 3rd party web sites which have their own pipelines. Dynamic content may be updated at intervals, as indicated in the last two lines of the figure, and connections tend to close in groups.

This web fetch method is illustrated schematically in Figure 3. We tend to build the following more detailed observations: 1) Connection to third-party servers. Fetching an object located on a thirdparty server needs the opening of a new TCP connection thereto server, over which the HTTP request is then sent. The TCP connection handshake introduces a delay (of a minimum of one RTT) and since the pattern of those delays is related to the web page content it can potentially assist in identifying the web page. 2) Pipelining of requests. Multiple objects located on the same server lead to many GET/POST requests being sent to that server, one after another. because of the dynamics of TCP congestion control, this burst of consecutive requests will have an effect on the timing of the response packets in a predictable manner that after once more will doubtless assist in identifying the web page. 3) Asynchronous requests. Dynamic content, e.g. pre-fetching via Ajax, can cause update requests to a server with large inter-arrival times which will potentially act as a web page signature. 4) Connection closing. once a web page fetch is completed, the associated TCP connections are closed. A FIN/ FINACK/ACK exchange closes each connection and this burst of packets can have quite distinctive timing that permits it to be known. Since the number of connections is related to the number of distinct locations where objects in the web page are stored, it changes between web pages.

Our aim is to use timing features like these, that vary relying upon the web page fetched, to create a timing signature which permits us to spot that web page is being fetched based on timing information solely.

IV. COMPARING SEQUENCES OF PACKET TIMESTAMPS

Suppose we've got two sequences of packet timestamps t := , i = 1, 2, • • • , n and t := , j = 1, 2, • • • , m. Note that for simplicity we tend to re-label the uplink packet indices to begin from one and to extend consecutively since none of our analysis can rely on this. Note also that the sequence lengths n and m aren't assumed to be the same. To proceed we want to outline an appropriate measure of the distance between such sequences.

A. Network Distortion of Timestamp Sequences

The packet stream determined throughout a web page fetch is affected by network events throughout the fetch. Changes in download rate (e.g. because of flows starting/finishing within the network) tend to stretch/compress the times between packets. Queuing within the network conjointly affects packet timing, with queued packets experiencing each larger delay and tending to be additional bunched together.

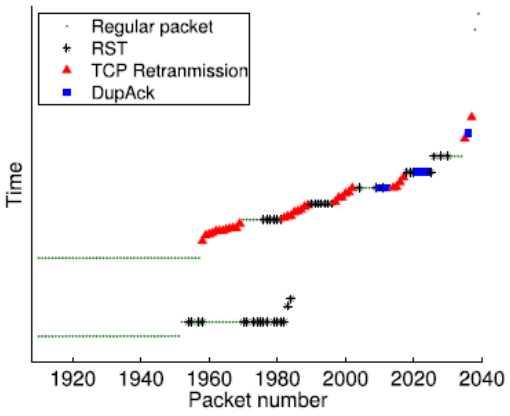

Fig. 4. Illustrating impact of changes in packet loss on the packet timestamp sequence. The bottom sequence shows the packet sequence at connection closing of a loss-free web fetch, while the top sequence shows the corresponding section from a different fetch of the same web page that was subject to packet loss and exhibits TCP retransmissions and DupACKs.

Link-layer retransmission on wire-less links includes a similar impact to queuing. Similarly to changes in download rate, the result is primarily to stretch/compress the times between packets. Packet loss introduces a "hole" within the packet stream where the packet have to be compelled to have arrived and also affects the timing of later packets due to the action of TCP congestion control (which reduces the send rate on packet loss) and retransmission of the lost packets. for instance, Figure 4 shows uplink measurements of packet retransmissions and duplicate ACKs at the end of two fetches of the same web page wherever it can be seen that these have the impact of stretching the packet sequence.

B. Derivative Dynamic Time Warping

Our interest is in a measure of the distance between packet sequences that is insensitive to the kinds of distortion introduced by the network, so the distance between packet streams t and t related to fetches of a similar web page at completely different times is measured as being small, and ideally the distance between fetches of various web pages is measured to be large. to the present finish we tend to use a variant of Dynamic Time warping (DTW) [9]. DTW aims to be insensitive to variations between sequences that are as a result of stretching/compressing of time so may be expected to at least partly accommodate the consequences of changes in download rate, queuing delay etc.

We outline a warping path p to be a sequence of pairs, , k = 1, 2, • • • , l with (pki , pkj) ∈ V := ×j j satisfying boundary conditions p1i = 1 =j p1 , pli = n, pl = m and step-wise constraints i (pki+1, pk+1) ∈ Vpki , pkj := ∩ , v ∈ ∩}, k = 1, • • • , l −1. That is, a warping path maps points from one timestamp sequence to another such that the beginning and finish points of the sequences match (due to the boundary conditions) and also the points are monotonically increasing (due to the step-wise constraints). This is often illustrated schematically in Figure 5, wherever the two timestamp sequences to be compared are indicated to the left and above the matrix and also the bold line indicates an example warping path. Let Pmnl ⊂ V l denote the set of all warping paths of length l associated with two timestamp sequences of length• :Pmnl → n and m respectively, and let Ct ,t ( )R be a cost function so that Ct ,t (p) is that the value Of warping path p ∈ Pmnl.

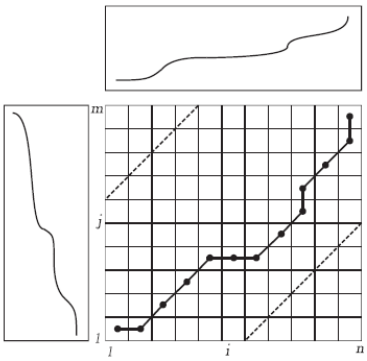

Fig. 5. Illustrating a warping path. The dashed lines indicate the warping window.

Our interest is within the minimum cost warping path. p∗ (t,t) ∈ arg min p∈ Pmnl Ct ,t (p). In DTW the cost operate has the separable form Ct ,t (p) = l where k=1 ct ,t (pk , pk) ct ,t : V → R, during which case optimal path p∗ (t,t) be efficiently found using the backward recursion, (pki , pkj) ∈ arg min Ck one + ct ,t (pi , p j) (1) (pi , p j)∈ Vk + Ck = Ck+1 + ct ,t (pki , pkj) (2) where Vk = (pi , p j) ∈ , k = l − 1, l − 2, • • • and initial condition Cl = ct ,t (n, m) When there's quite one optimal solution at step (1), we choose (pki , pkj) uniformly at random from amongst them.

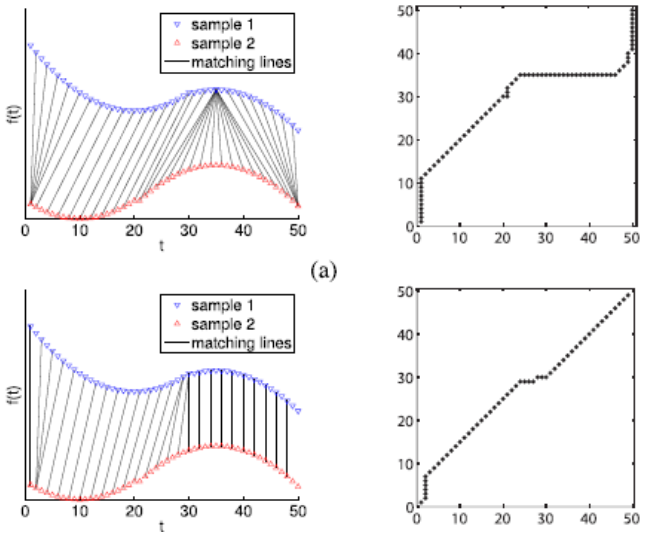

Fig. 6. Example DTW alignment and warping paths between two sequences vs cost function *ct ,t* used, window *w* = 0.1.

In this example the length *l* of the warping path is 73 when a Euclidean cost is used and 54 with the derivative cost. (a) Euclidean cost. (b) Derivative cost. A common selection of element-wise cost is that the Euclidean norm ct ,t (pi , p j) = (tpi − tp j)2. However, in our data we found that this cost can result in all the elements of one sequence that are beyond the last element of the other sequence being matched to that single element. For this reason and also to improve robustness to noise on the timestamp values (in addition to misalignment of their indices), following [9] we instead use the following element-wise cost ct ,t (pi , p j) = (Dt (pi) − Dt (p j))2 (3) where Dt (i) = (ti −ti−)+(ti+ −ti−) , i −= max i one, one} and i + = 2 t . Observe that Dt (i) is akin to the derivative of sequence t at index i .

Further, we constrain the warping path to remain within windowing distance w of the diagonal (i.e. within the dashed lines indicated on Figure 5) by setting C(p) = +∞ for paths p ∈ Pmnl for which | pki − pkj | > max, |m − n|} for any k ∈ . Figure 6b illustrates the alignment of points between two sequences obtained using this approach and for comparison Figure 6a shows the corresponding result when using Euclidean cost. The figure shows the warping paths on the right-hand side and an alternative visualization of the mapping between points in the sequences on the left-hand side. Observe that when Euclidean cost is used the warping path tends to assign many points on one curve to a single point on the other curve. As noted in [9] this can be best-known to be a feature of Euclidean cost. As compared, use of the derivative distance tends to mitigate this impact and choose a warping path with fewer horizontal and vertical sections.

C. F -Distance Measure

Given two timestamp sequences, the warping path could be a mapping between them. With relevancy Figure 5, sections of the warping path that lie parallel to the diagonal correspond to intervals over which the two sequences are well matched. Sections of the warping path that are parallel to the x- or y-axes correspond to intervals over that the two sequences are poorly matched. This suggests using the fraction of the overall warping path which is parallel to the x- or y-axes as a distance measure, which we tend to refer to as the F-distance. In more detail, let p = , k = 1, • • • , l be a derivative DTW warping path relating timestamp sequences t and t , obtained as described in the previous section.

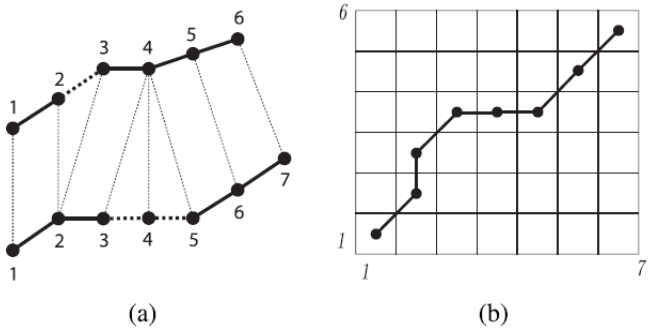

Fig. 7. Illustrating method for calculating the *F*-distance between two timestamp sequences. (a) Sequence alignment. (b) Warping path.

We partition the warping path into a sequence of sub paths among each of which either pki or pkj remain constant and that we count the sub paths which are longer than one. as an example, for the setup shown in Figure 7 there are five sub paths: (1, 1); (2, 2), (2, 3); (3, 4), (4, 4), (5, 4); (6, 5); (7, 6). two of those sub paths contains more than one pair of points, namely (2, 2), (2, 3) and (3, 4), (4, 4), (5, 4), and these correspond, respectively, to the vertical section and the horizontal section on the corresponding warping path shown in Figure 7b.

Formally, define κ 1 := 0 < κ 2 • • • i – I := l such < < κr 1 < κr that for each s = 1, • • • , r − 1 (i) either pk1 = pk2 ∀ k1, k2 ∈ {κs + jj∀ k1, k2 ∈ {κs + 1, • • • , κs+1} 1, • • • , κs+1} or pk1 = pk2 and (ii) either κs+1 =l or condition (i) is violated for some k , k ∈ {κs I • • • j s+1 + 1 } i.e. each subsequence is maximal. , κ Note that pk = pk for all k = 1, • • • , l (due to warping path step-wise constraints) and so in condition (i) it's not possible for both pki and pkj to be constant. We are now's a position to define the F-distance measure between timestamp sequences t and t , namely: s κs+1 – κs φ (t , t) := κs+1 −κs >1 (4) n + m where κs , s = 1, • • • , r are the constant subsequences in minimal warping path p∗ (t , t). It can be seen that φ (p) takes values in interval [0,1], and is 0 when sequences t and t are identical (in which case the warping path p lies on the diagonal in Figure 5). For the example in Figure 7 the F-distance φ (p) is (2 + 3)/13 = 0.385.

V. DE-ANONYMISING WEB FETCHES OVER AN ETHERNET TUNNEL

In this section we have a tendency to present measurements of web page queries carried out over an Ethernet tunnel and evaluate the accuracy with which the web page being fetched can be inferred using only packet timing information. The whole project together with codes, scripts and datasets for all measurement campaigns is available at [4]. The primary dataset consists of home pages of each of the top Irish health, financial and legal web sites as ranked by www.alexa.com under its Regional/Europe/Ireland category in November 2014. We tend to prune the pages that fail to load and then for each of the top a hundred sites we carry out a hundred fetches of the index page yielding a complete of 10,000 individual web page fetches in a dataset.

For comparison we collected two such datasets, one where the pages of each web site are fetched consecutively over an hour and a second where the pages are fetched each hour over a period of 5 days. In these datasets the browser cache is flushed between every fetch in order that the browser always starts in a very recent state. Additionally, a 3rd dataset was collected consisting of the same 10,000 web fetches however currently without flushing of the browser cache between fetches. the web pages were fetched over a period spanning November 2014 to January 2015. A watir-web driver script on Firefox 36.0 was used to perform the web page fetches and tcpdump to record the timestamps and direction (uplink/downlink) of all packets traversing the tunnel although solely packet timestamps on the uplink were truly used.

A. Hardware/Software Setup

The network setup consists of a client that routes traffic to the net over a gigabit Ethernet LAN. The client machine is a Sony VGN-Z11MN laptop with an Intel core 2 duo 2.26GHz processor and 4GB of memory. It's running Ubuntu Linux 14.04 LTS Precise.

B. Classifying Measured Timestamp Sequences

We use the F-distance measure φ (•, •) described in Section IV to check measured uplink timestamp sequences, with windowing parameter w = 0.2 unless otherwise stated. Figure 8 shows example scatter plots obtained using this distance measure. in more detail, from the set Ti of measured timestamp sequences for the i-th web site we choose a sequence t i that minimizes t ∈ Ti φ (t , t i) then use ti as the exemplar for the i -th web page.

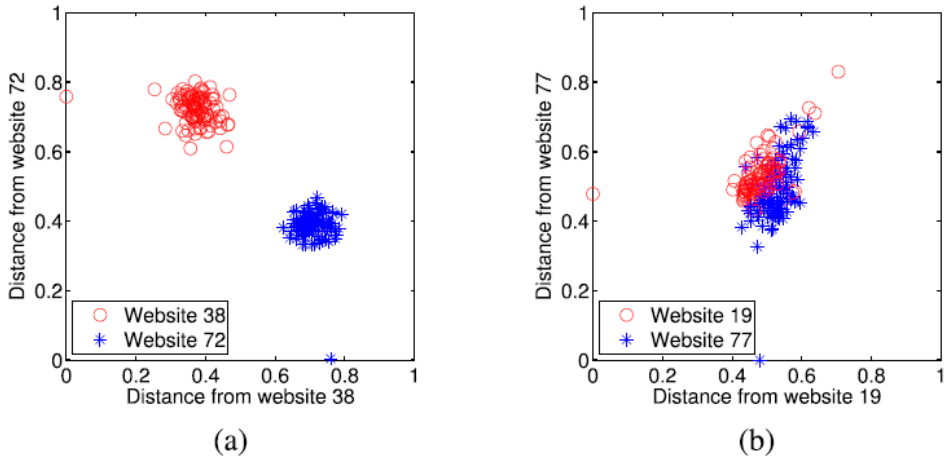

Fig. 8. Scatter plots for 4 different web pages using *F*distance measure *φ*. In (a) two relatively distinct web pages are compared while the web pages in (b) are relatively similar.

In Figure 8 we then plot φ (t , t i) for each of the timestamp sequences t measured for web page i and also for timestamp sequences measured for another web page. in the example in Figure 8a it is seen that the distance measure is so effective at separating the measured timestamp sequences of the two web pages thought of into distinct clusters, therefore potentially providing a basis for accurately classifying timestamp sequences by web page. Figure 8b shows an example of a scatter plot where the separation between the 2 web pages is less distinct and so classification are often expected to be less reliable.

As we'll see, examples of this latter sort prove to be fairly rare. We considered two approaches for using φ (•, •) to classify timestamp sequences: K –Nearest Neighbors and Naive Bayes Classification.

1) K-Nearest Neighbours:

In this method, for each web page i we tend to sort the measured timestamp sequences t ∈ Ti used for training in ascending order of sum-distance and select the top three to use as exemplars to represent this web page. Once presented with a new timestamp sequence, its distance to the exemplars for all of the training web pages is calculated and these distances are sorted in ascending order. Classification is then carried out by majority vote amongst the highest K matches.

2) Naive Bayes Classifier:

For every web page i from the measured timestamp sequences Ti used for training we tend to choose ti ∈ arg min t ∈Ti φ (t , t) (in addition we tend to conjointly take into account selecting ti to minimize the variance of the distance φ, see below) and then fit a Beta distribution to the empirical distribution of φ (t , t i) for t ∈ Ti . Let pi (•) denote the probability distribution obtained in way. once presented with anew timestamp sequence t , we tend to calculate the probability pi (t) of this sequence belonging to web page i and choose the web page for which this probability is greatest.

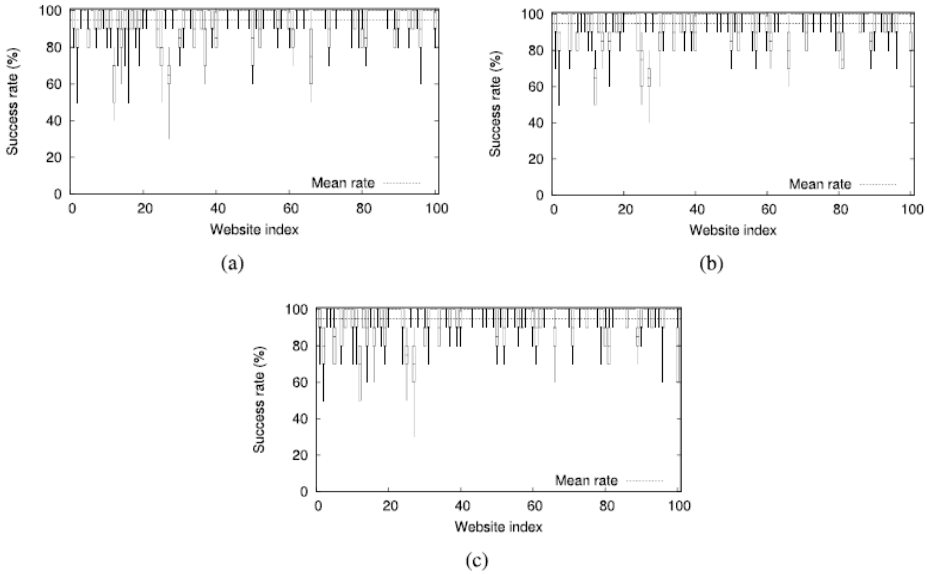

Fig. 9. *K* – Nearest Neighbours classification performance, no browser caching. (a) *K* = 1. (b) *K* = 3. (c) *K* = 5.

C. Experimental Results

We begin by presenting results for the dataset wherever pages are fetched consecutively and therefore the browser cache is flushed between fetches. Figure 9 details the measured classification accuracy using the K –NN approach, for numerous values of K . We tend to use 10-fold cross validation, where the {10|the ten}0 samples of every site are divided into ten random subsets and for every subset we tend to use the remaining ninety samples as the coaching knowledge to search out the exemplars and use the 10 samples within the subset as the validation knowledge. The rates for these ten subsets for each site are summarized and displayed in the figure. Each of the boxes indicate the twenty fifth, five hundredth and Seventy fifth quartiles and therefore the lines indicate the maximum and minimum values. The mean success rates for K = 1, K = 3 and K = 5 are 95.01%, 94.97% and 94.98% respectively. These results for uplink traffic compares to a most success rate of 92.5% when using packet timestamps on the downlink for the classification, indicating that use of uplink or downlink timestamps has very little impact on the performance of this classification attack. The results also are compared for a subset of fifty websites selected randomly from the present 100, see Table I, that conjointly confirms that the result of population size is minor. For comparison, the success rates once web pages are fetched hourly over five days are 90.88%, 90.72% and 90.74%. Observe that there's a little (about 5%) reduction in success rate, which we tend to assume is related to the time-varying nature of a number of the web sites. We tend to discuss the impact of content and speed variability on the performance in Section VII.

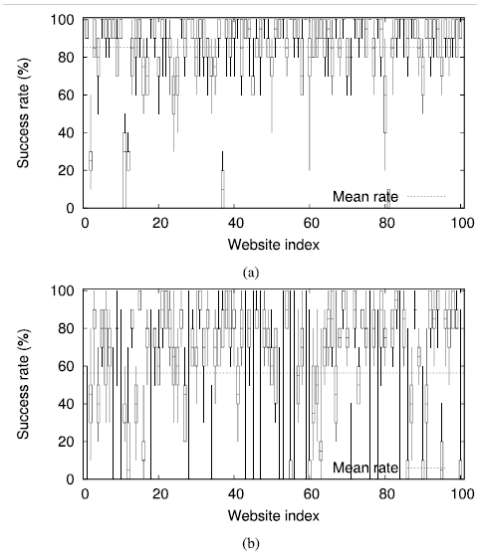

Fig. 10. Naive Bayes classification performance, no browser caching. (a) Minimum mean. (b) Minimum variance.

Figure 10 plots the corresponding results obtained using the naive Bayes approach. Performance is calculated once the exemplar for each web page is chosen to minimize the mean and therefore the variance of the distance. The mean success rates are 85.2% and 56.3% respectively. Since the performance is systematically worse than that of the K –NN classifier we tend to don't take into account the naive Bayes approach further in the rest of the paper.

D. Commonplace vs. Cached:

Totally different Versions of Same web page on first visiting a new web page a browser requests all of the objects that form the web page. However, on subsequent visits several objects could also be cached e.g. images, css and js files, etc. in the Mozilla browser, once the address of a web page is just entered again shortly after the complete page is fetched, since the cached copy of an object has not nevertheless expired the cached copy are used when rendering the web page and it'll not be fetched over the network by the browser. Bbut the browser are often forced to reload the web page by pressing F5 where it then sends a request for the objects and therefore the server might either come back an abbreviated NOT modified response if the cached object is actually still recent or return the complete object if it's modified.

Ultimately a full refresh are often induced by pressing Ctrl+F5 that requests for the complete version of the web page as if no object is cached before. Hence, the network traffic generated by a visit to a web page could differ significantly counting on whether or not it's been visited recently (so the cache is fresh) or not.

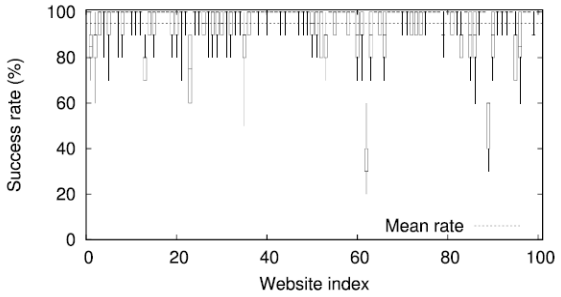

Fig. 11. *K*-Nearest Neighbour classification performance, with browser caching using 3 exemplars for each site. $K = 5$.

Classification of cached web pages are often expected to be tougher than for non-cached pages since there's less network traffic and so less knowledge upon which to base the classification decision. Figure 11 presents the measured classification accuracy once browser caching is enabled. This knowledge is for the case where requests that reply with NOT modified use the cached content, which is perhaps the most common form of caching used in practice. It can be seen that no matter the small size of the network traffic in this setup, the overall success rate for identifying web pages remains in far more than 95th.

E. Web pages outside the training Set

The experiments within the previous 2 sections are conducted with the idea that the adversary is aware of that the web page that the user has visited is among the set of web pages for which training knowledge has been collected. when this assumption need not hold, i.e. the user might need fetched a web page outside of the adversary's training info, then we can use the following approach to first classify whether or not a measured packet timestamp sequence t is associated with an internet web site in the training set or not. Recall that, as mentioned in Section V-B1, for every web page i within the training set we've got three exemplar packet timestamp sequences that are used for K –Nearest Neighbor classification.

Given a packet timestamp sequence t we tend to use K -Nearest Neighbor classification to estimate the closest web page w(t) among the training set and let Fmin (t) denote the minimum F-distance between the exemplars for this web page and therefore the measured timestamp sequence. We can then use this value as the basis for a simple classifier. Namely, when Fmin (t) is larger than a specified threshold (which might rely upon w(t)) then we tend to estimate t as lying outside the training set, and when Fmin (t) is below the threshold then we tend to estimate t as lying within the training set. It remains to select an appropriate threshold for every web page within the training set.

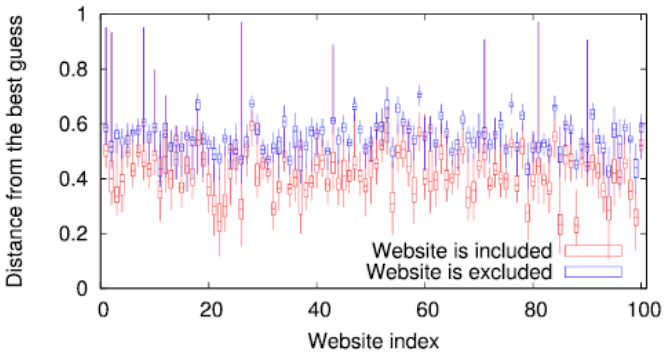

Fig. 12. Distribution of the *F*-distance between the measured packet timestamp sequences in the training dataset and the exemplar packet sequences for the best guess.

Data is shown for when sequences of each web site are within the training dataset and for when they are removed. Ethernet channel, no browser caching. For every timestamp sequence t within the training set Figure 12 plots the distribution of Fmin (t) vs the index of the web site that t is measured. This figure could be a box and whiskers plot with the min, max and quartiles shown. for each site we tend to then take away its information from the training set and repeat the calculation. The distribution of those values is also shown in Figure 12. It can be seen that, unsurprisingly, the F-distance is Systematically higher when a web site is excluded from the training set. We tend to choose the threshold for classification to try to separate these 2 sets of value. Namely, we tend to take the average of the x percentile of the lower values and therefore the (100−x) percentile of the higher values as our threshold, where $0 \leq x \leq 100$ could be a design parameter.

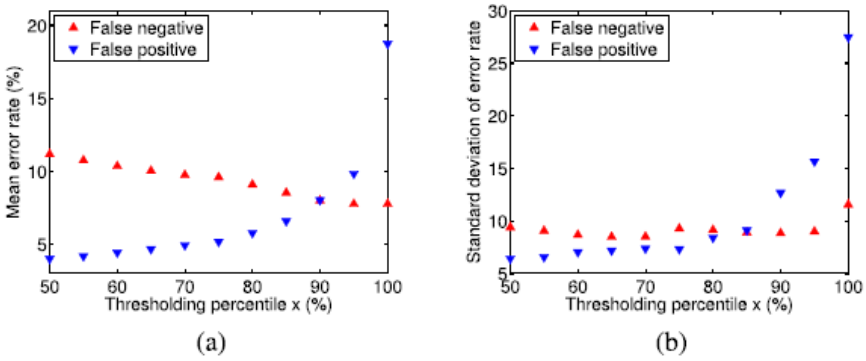

Fig. 13. Mean and standard deviation of false negative and false positive error rates vs the choice of F distance threshold (specified via design parameter x). (a) Mean, (b) Standard Deviation.

The classification error rate vs the threshold parameter x used is shown in Figure 13a. two error rates are shown, first of all the fraction of web pages that are out with the training set however that are classified as lying within it (which we tend to refer to during this section as false positives) and secondly the fraction of web pages that are among the training set however that are classified as lying out with it (which we tend to refer to as false negatives). The standard deviations of those error rates across the web pages are also shown in Figure 13b. It can be seen that thresholding with x = 90 yields equal error false negative and false positive rates of about 8.0%, that is near the complement of the reported success rate reported within the preceding section.

VI. MEASUREMENT RESULTS FOR OTHER CHANNELS

In this section we tend to extend consideration from Ethernet to a number of various network channels. Namely, we tend to take into account packet timestamp measurements taken from a commercial femtocell carrying cellular wireless traffic, from a time-slotted wired UDP channel (of interest as a potential defence against timing analysis) and from the first hop (i.e. between the client and the Tor gateway) of a Tor channel. similar to before, in every case we tend to collected packet timestamp knowledge for a hundred fetches of the home pages of each of the top one hundred Irish health, financial and legal websites as ranked by www.alexa.com.

A. Femtocell Traffic

A femtocell is an eNode B cellular base station with atiny low physical footprint (similar to a wifi access point) and restricted cell size (typically about 30m radius). it's intended to enhance cellular coverage indoors, filling in coverage holes and improving download rates, whereas additionally offloading traffic from the macro cell network. Wired backhaul to the cellular operators network is via a user supplied network connection e.g. a home DSL line. Since femtocells are usually user installed, physical access to the backhaul connection is straightforward and it's an easy matter to route backhaul traffic via a sniffer. Mobile operators are, of course, responsive to this and backhaul traffic is thus secured via use of an IPSec encrypted tunnel. Within the setup thought of here, the femtocell backhaul is over a university gigabit Ethernet connection and that we used tcpdump to log packets passing over this link.

1) Hardware/Software Setup:

The client computer is the same Sony laptop used for the Ethernet measurements. It currently uses a Huawei K3770 HSPA USB Broadband dongle to connect wirelessly to the internet via a Femotcell. The femtocell may be a commercial Alcatel-Lucent 9361 Home Cell V2-V device. The femtocell wired backhaul is connected to a campus network via a NetGear en 108 TP Ethernet hub. A monitor computer that is running on a AMD Athlone 64 X2 dual Core Proc 5000 + cpu and 4GB memory is also connected to the current hub and logs all packets. The client and monitor computers both run Ubuntu Linux 14.04 LTS Precise.

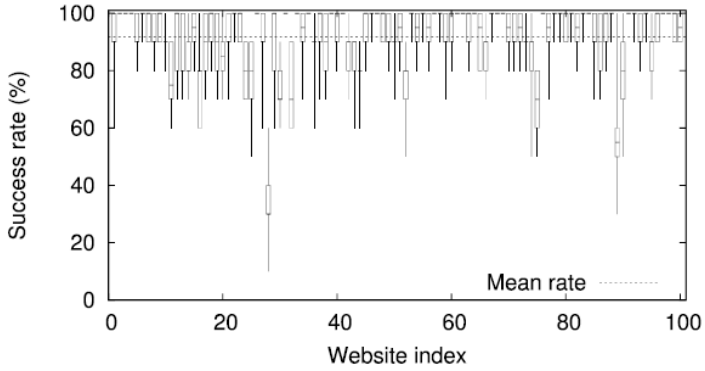

Fig. 14. Femtocell *K* -Nearest Neighbours classification performance, no browser caching, *K* = 5.

2) Results:

In contrast to the comparatively clean Ethernet channel thought of in Section V-C, we tend to found that traffic passing over the wireless femtocell link is often distorted by factors like wireless and cellular noise, encoding/decoding delays, cellular control plane traffic etc. These distortions usually seem as shifts along the x -axis of the packet timestamp patterns and/or as delays within the y-axis. The measured performance employing a K -NN classifier using three exemplars for every website and K = 5 is shown in Figure 14. The mean success rate is 91.8%, that compares with the mean success rate of ninety fifth observed in Section VC when employing a clean Ethernet channel. It can be seen that use of the wireless channel tends to scale back the classification accuracy, as might be expected due to the additional loss/delay over the wireless hop. However, the reduction in accuracy is minor.

B. Time Slotted UDP Tunnel

We developed a custom tunnel using ip tables, net filter and net filter-queue. The tunnel trans-ports packets over a UDP channel in a time slotted fashion and therefore the slot size is a configurable parameter.

1) Hardware/Software Setup:

The experimental setup is identical to that used in Section V apart from the employment of a customized tunnel. On the client computer all web traffic is captured using the OUTPUT net filter hook, encapsulated into UDP packets and sent to a server at the other side of the tunnel. The server, which has an AMD Athlone 64 X2 dual Core Proc 5000+ and 4GB memory, fetches these UDP packets using the PREROUTING hook, extracts the payload and sends them by via the FORWARD hook to the outgoing Ethernet interface. Similarly, incoming packets from the internet are encapsulated into UDP packets via FORWARD hook on the server and sent to the client that captures them using the PREROUTING hook, extracts the payload and forwards this to the application layer.

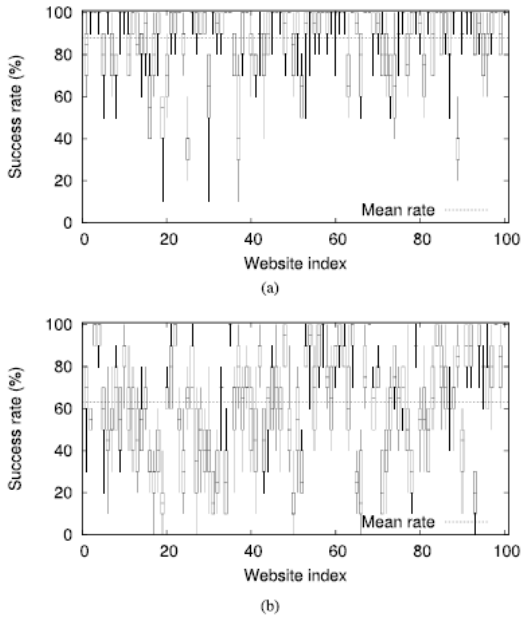

Fig. 15. Time-slotted tunnel *K* -Nearest Neighbours classification performance, no browser caching, *K* = 5.(a) Slot size: 1ms. (b) Slot size: 10ms.

2) Results:

Figure 15 shows the measured performance employing a K-NN classifier where three exemplars are chosen from each website and K = 5. The overall success rate is 880 yards when the tunnel slot size is 1ms and sixty three once the tunnel slot size is increased to 10ms. we also considered slot sizes larger than 10ms, but since we tend to found such large slot sizes tended adversely have an effect on browser performance (and therefore would likely be problematic in practice) we tend to don't include them here. This performance compares with successful rate of ninety fifth over a plain Ethernet tunnel. As may be expected, time-slotting decreases the classification success rate since it adds timing "noise". However, even with a relatively giant slot size of 10ms the impact on performance isn't proportional to the sacrifice we tend to make in terms of delay and throughput (with such an oversized slot size we are capping the downlink throughput to 150KB/s). This approach thus seems to be unappealing as a sensible defence against the timing based attack considered here. In fact additional refined forms of defence could also be more practical, however we tend to leave thought of these to future work as they seemingly involve advanced trade-offs between network performance and resistance to attack that we lack space to address here.

C. Tor Network

In this section we tend to consider measurements of web page queries over the Tor network. Tor is an overlay network of tunnels that aims to enhance privacy and security on the net.

1) Hardware/Software Setup:

The experimental setup is that the same as in Section V except that the traffic from the client browser, Mozilla Firefox 36.0 is proxified over Tor v0.2.5.11. Note that we tend to conjointly explored use of the Tor browser however found that a major set of the online sites failed to load, timed out or needed a CAPTCHA to be solved or every page fetch that created complications when scripting fetches. We tend to conjointly investigated using Firefox with Tor pluggable transports (such as obfs4 etc.) however we tend to found that using these add-ons had a large impact on delay such that most websites fail to load even after five minutes. As before, the browser cache is flushed between fetches.

2) Randomized Routing:

Tor uses randomized routing of traffic over its overlay network in an attempt to make linking of network activity between source and destination harder. It is expected that rerouting can have a major impact on the timestamp sequence measured throughout a web fetch since changes in path propagation have a direct impact on the time between an outgoing request and receipt of the corresponding server response, and also impact TCP dynamics since congestion window growth slows with increasing RTT. variations in loss rate, queuing delay etc. on completely different routes also are doubtless to impact measured timestamp sequences.

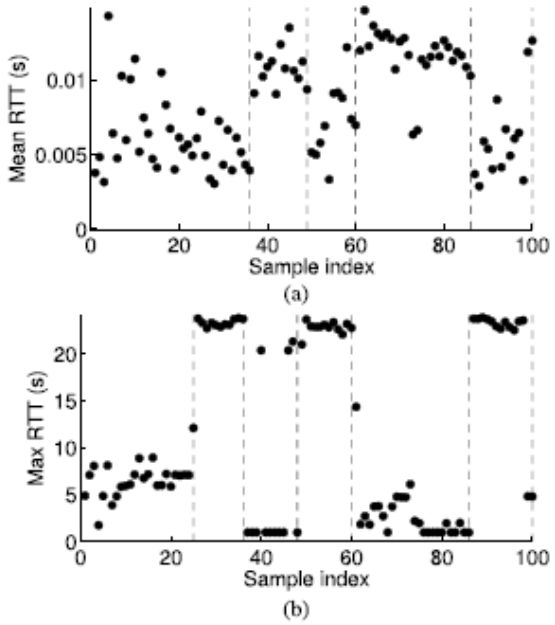

Fig. 16. Mean and max RTTs measured during 100 fetches of the web page www.medicalcouncil.ie. Changes due to Tor rerouting are evident.

The max RTT in (b) is in fact the idle time between when the last packet is received until the browser is closed, hence why it is significantly larger than the mean RTT plotted in (a). (a) Mean RTT for packets of each sample. (b) Max RTT for packets of each sample. The impact of Tor rerouting on measured RTT is illustrated in Figure 16,that plots the mean and max delay between sending of a TCP data packet and receipt - of the corresponding TCP ACK for repeated fetches of the same web page (although his data isn't available to an attacker, in our tests it is in fact accessible for validation purposes).

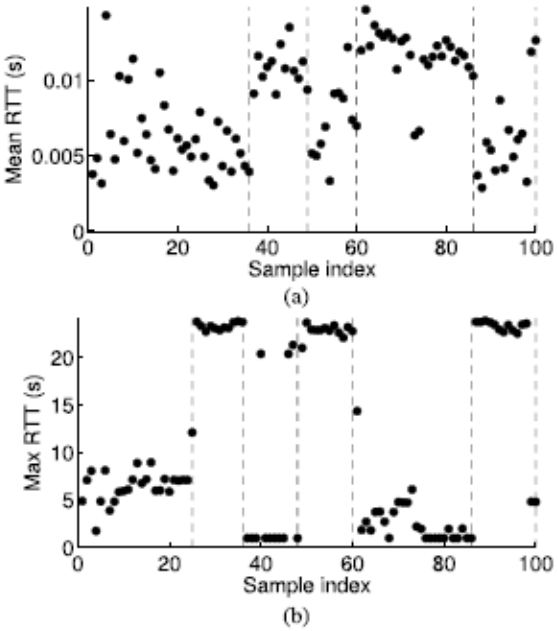

Fig. 17. Time traces of uplink traffic measured when fetching www.medicalcouncil.ie. Measurements are shown both when using vanilla Firefox and when using Firefox with the Tor plugin.

Abrupt, substantial changes within the mean RTT are evident, particularly in Figure 16b. These changes persist for a period of time as Tor solely performs rerouting periodically. Figure 17 illustrates the impact of Tor on the packet timestamps measured throughout a web page fetch.

3) Results:

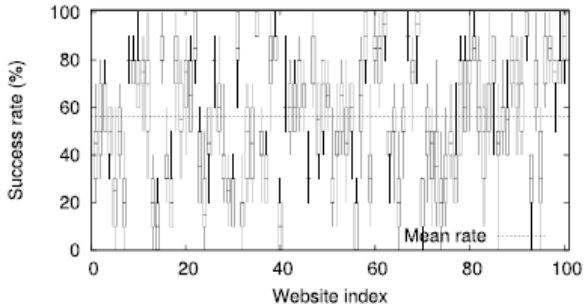

Fig. 18. Tor network *K* -Nearest Neighbours classification performance, no browser caching, *K* = 5.

Figure 18 details the measured classification accuracy using the K –NN approach, where three exemplars are chosen from every web site and a window size of w = 0.2 is used to accommodate the warping between samples. The mean success rate is 56.2% that compares with the mean success rate of 95.0% when using a clean Ethernet channel. As may well be expected, use of the Tor network considerably reduces classification accuracy. However, the success rate of 56.2% compares with a baseline success rate of a hundred and twenty fifth for a random classifier over one hundred websites so still is probably going to represent a significant compromise in privacy. We tend to note also that this compares favorable with the 54.6% rate reported by Panchenko et al in [14] against Tor traffic using packet size and direction information.

D. Alternative proposed Channels

A number of alternative channels are proposed within the literature as a defence against traffic analysis attacks. Wright et al. [18] suggest a traffic morphing technique that maps the packet sizes of 1 website to the packet distribution of another site. This defence fails to beat the attack considered here since it makes use solely of timing info and doesn't use packet size info. This is also the case for all of the packet –size based defences proposed within the HTTPOS scheme introduced in [12]. a potential defence against timing attacks is to change the packet timing pat-tern by delaying transmissions. However, though this might be expected to counter timing-based attacks like that considered here such defences also will have a control on delay. for instance, BuFLO introduced in [3] is analogous to the time slotting method that we consider above and that seems to be impractical given its substantial impact on delay and bandwidth, with 190th bandwidth overhead reported in [16].

VII. EFFECT OF LINK SPEED AND CONTENT CHANGE ON CLASSIFICATION PERFORMANCE

By looking closely at performance of websites, it may be seen that the total mean success rate obtained in every measurement campaign isn't monotone amongst individual websites. In this section, we tend to investigate possible reasons behind the poor performance of certain websites. we tend to use the same Ethernet dataset from Section V-C where samples are fetched hourly over five days. The study of different situations like femtocell, cached etc. provides similar results. 1) Network Speed. The link speed between the client and every web server varies from a web site to another. it's conjointly completely different between samples of the same page. To investigate the impact of network speed on the classification performance, we calculated peak downlink speed throughout every fetch (the results for uplink and uplink + downlink speed is similar). Then so as to compare the metrics, values for samples of every page are normalized and their variance is evaluated.

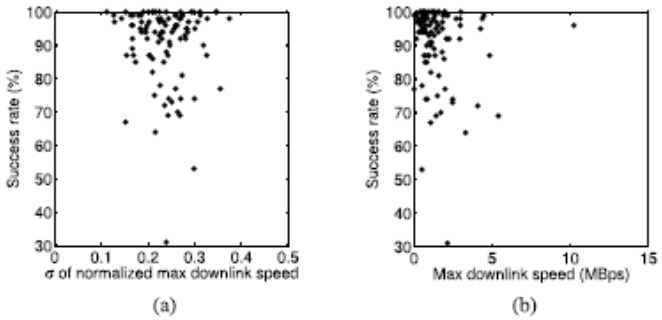

Fig. 19. Scatter plot of max link speed standard deviation and median against success rate. Samples are taken hourly for 5 days over ethernet channel.(a) Standard deviation. (b) Median.

TABLE I
SUMMARY OF THE MEASURED SUCCESS RATE OF THE PROPOSED ATTACK REPORTED HERE. DATA IS SHOWN FOR DIFFERENT NUMBERS OF EXEMPLARS, DIFFERENT POPULATION SIZES AND DIFFERENT VALUES OF K IN THE K-NEAREST NEIGHBOURS METHOD. IN ALL CASES THE SAMPLES OF EACH WEB SITE ARE FETCHED CONSECUTIVELY WITHIN AN HOUR EXCEPT FOR (*) WHERE A SAMPLE IS TAKEN EACH HOUR FOR 5 DAYS

Channel		Number of Exemplars	Database size	K			
				1	3	5	7
Ethernet		5	100	95.27%	95.65%	95.86%	95.74%
		3	100	95.01%	94.97%	94.98%	-
		3	100*	90.88%	90.72%	90.74%	-
		1	100	93.37%	-	-	-
		3	50	97.16%	97.18%	97.04%	-
Ethernet (Downlink)		3	100	92.47%	91.64%	90.79%	-
Cached		3	100	95.88%	95.30%	95%	-
Slotted	1ms	3	100	89.23%	88.25%	87.98%	-
	10ms	3	100	63.73%	61.40%	63.35%	-
Femtocell		3	100	92.60%	91.80%	91.83%	-
Tor		3	100	58.44%	56.18%	56.2%	-

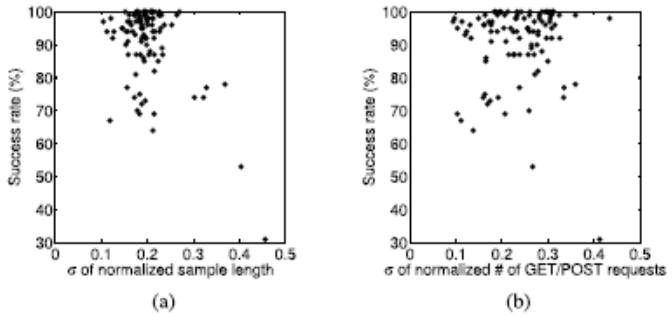

Fig. 20. Scatter plot of sample length and GET/POST request count standard deviation against success rate. Samples are taken hourly for 5 days over Ethernet channel. (a) Sample length. (b) GET/POST count.

Figure 19a illustrates the scatter plot of normalized standard deviation of link speed against success rate of every web site. It may be seen there's no strong correlation between these two metrics that will recommend that a web site with more variable link speed ought to result a lower success rate. Similar comparison is additionally studied with median speed for every web site (Figure 19b) to indicate that having an overall faster link speed doesn't guarantee a poor classification performance. 2) Sample Length and GET/POST Requests Count. for every web site we tend to plot the standard deviation of the normalized number of uplink packets (a measure of the variability of the web page over time) and also the corresponding success rate (see Figure 20a). The results for uplink and uplink + downlink are similar. We tend to conjointly provided the same plot for max number of GET/POST requests for every web site (Figure 20b). It may be seen that, there's no strong correlation between the these metrics and success rates that is suggestive that the classification attack is fairly insensitive to variability of web page content over time. 3) IP Connections, Active TCP ports. so as to research the robustness of the attack against parallel connections, for every web site we tend to plot the median number of serving ip connections and active TCP ports against their corresponding success rates.

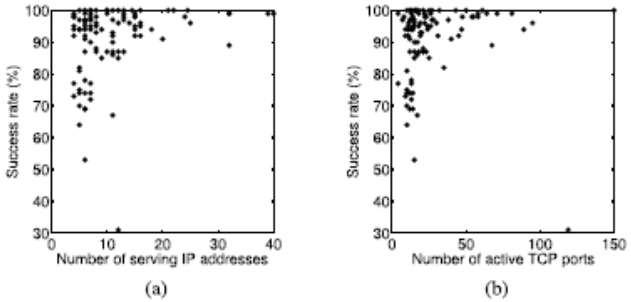

Fig. 21. Scatter plot of median open IP connections and active ports count against success rate. Samples are taken hourly for 5 days over ethernet channel. (a) Open IP connections. (b) Active TCP ports.

As illustrated in Figures 21a and 21b, again there's no clear correlation between mentioned metrics that is suggestive that the number of active IPs/ports for every web site, that represents the number of parallel connections, has no effect on the performance of our proposed attack.The above results suggest that there's no strong correlation between the performance of our attack and link speed, little content modification and number of parallel connections.

However the selections of exemplars are essential to the performance of the attack. In particular when the content modification is quite a threshold, the distinction between samples will now not be ignored by the attack. An example of this misbehavior may be seen for web site #10 within the measurement campaign considered during this section, where two completely different versions of the page were observed throughout the experiment.

In result, one exemplar represents one version while two others represent another version of the page. This causes K -NN method to fail collecting enough votes for a successful classification that successively ends up in successful rate of 31st. To overcome this issue, separate sets of exemplars are needed to represent every version of a web page so as to successfully classify future samples.

VIII. FINDING A WEB PAGE WITHIN A SEQUENCE OF WEB REQUESTS

In the experiments presented so far we've assumed that among the observed packet timestamp stream the boundaries between totally different web fetches are best-known. This can be most likely a reasonable assumption on gently loaded links where the link is often idle between web fetches. However, not only might this assumption be

less applicable on more heavily loaded links however it conjointly permits for a comparatively simple means that of defence, namely insertion of dummy packets to obscure the boundaries between web fetches. During this section we have a tendency to thus extend consideration to links where web fetches are carried out in a back to back fashion such that the boundaries between web fetches can not be simply known. The basic idea is to sweep through a measured stream of packet timestamps making an attempt to match sections of it against the timing signature of a web page of interest. This exploits the very fact that our timing-only attack doesn't basically depend upon information of the start/end times of web fetch (unlike previous approaches that use packet counts to classify web pages).

In more detail, to locate a target web page within a stream of packet timestamps we tend to first choose three measured packet timestamp sequences for that web page to act as exemplars (as previously). Then, we tend to sweep through the stream of timestamps in steps of ten packets, extract a section of the stream of the same length as every exemplar (plus ten to cover the step size) and calculate the distance between the section and the exemplar. Once sweeping through the complete stream we tend to choose the location within the stream with least distance from the exemplars because the likely location of the target web page within the stream. Whereas this process assumes that the target web page is present within the packet stream, using a similar approach to that in Section V-E we could extend this approach to decide whether or not the web page is present by appropriately thresholding the distance (when the measured least distance is above the threshold, the page is judged to not be present in the stream).

A. Results

We constructed a test dataset as follows. For every run we tend to choose one of the 100 websites to be the target. We then uniformly at random pick up to four different websites from the remaining websites. The chosen websites are then permuted randomly and fetched one after another with a pause after every fetch acting as a "thinking period". The utmost time allowed for every fetch to finish is 25 seconds i.e the length of every pause is chosen uniformly at random from 5-25 seconds. Repeating this for all websites within the dataset, we tend to created one hundred test runs.

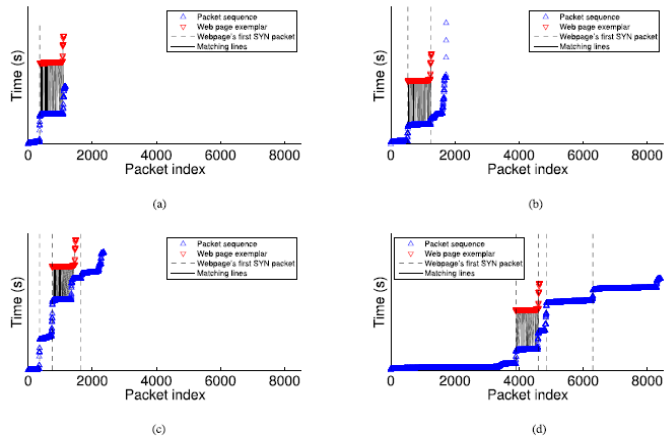

Fig. 22. Illustrating locating of a web page within a packet stream.

The page www.iscp.ie shown in red triangles is an example of a web page which is successfully located among 2, 3, 4 and 5 consecutive web fetches. The vertical lines show the first SYN packet of each web page. (a) Two consecutive web fetches. (b) Three consecutive web fetches. (c) Four consecutive web fetches. (d) Five consecutive web fetches. Using the classification approach described above we tried to spot the location within every packet stream. Figure 22 presents four samples of this, showing the position within a stream with least distance from the exemplars of a target web page. The success rate results for streams of 2-5 websites are summarized in Table II.

TABLE II
SUCCESS RATES OF LOCATING WEB PAGES AMONG 2-5 FETCHES

No. of consecutive pages	2	3	4	5
Success rate	82%	80%	66%	64%

With this approach we tend to achieved a maximum success rate of 82 for locating the target web page within every packet stream within a position error of w.ls packets, where w is the window size at which DTW operates (0.2 in our setting) and ls is the average length of the three exemplars that are determined for every web site s separately. Given the limited info being used, this is a remarkably high success rate and indicates the ability of the timing-only attack. However, it is seen that the success rate starts to lower as the range of consecutive fetches grows which ends up in a extended packet stream that may probably include similar patterns to the target web page. Furthermore web pages with shorter length are less doubtless to be located properly due to their shorter signatures that are more probably to appear within the middle of a larger net trace.

IX. SUMMARY AND CONCLUSIONS

We introduce an attack against encrypted web traffic that makes use solely of packet timing information on the uplink. Additionally, in contrast to existing approaches this timing-only attack doesn't need knowledge of the start/end of web fetches and so is effective against traffic streams. We tend to demonstrate the effectiveness of the attack against both wired and wireless traffic, consistently achieving mean success rates in more than 90th. Table I summarizes our measurements of the success rate of the attack over a variety of network conditions. Study of downlink and a preliminary study of uplink + downlink traffic recommend very little difference from uplink results presented in this paper, given timing patterns of uplink and downlink are powerfully correlative.

Moreover, the proposed attack proves to be robust against totally different link speed, totally different range of parallel connections and little content modification, having the ability to keep up overall success rate of ninety one for measurements collected over a course of five days. However the threshold that the attack remains resilient to content modification is to be studied. We tend to leave any investigation of those matters for future work. Since this attack solely makes use of packet timing information it's run proof to existing packet padding defences. We tend to show that time slotting is also insufficient to prevent the attack from achieving a high success rate, even once relatively large time slots are used (which could be expected to significantly distort packet timing information).

Similarly, randomized routing as used in Tor is also not effective. more subtle forms of defence could also be more practical, however we tend to leave consideration of those to future work as they probably involve complicated trade-offs between network performance (e.g. increased delay and/or reduced bandwidth) and resistance to attack that warrant a lot of detailed study than is possible here. In addition to being of interest in its own right, by highlighting deficiencies in existing defences this timing only attack points to areas where it might be useful for VPN designers to focus further attention.

REFERENCES

[1]. G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted HTTP streams," in *Privacy Enhancing Tech-nologies* (Lecture Notes in Computer Science), vol. 3856, G. Danezis and D. Martin, Eds. Berlin, Germany: Springer, 2006, pp. 1–11.

[2]. X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a Distance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2012, pp. 605–616.

[3]. 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 Symp. Secur. Privacy (SP)*, May 2012, pp. 332–346.

[4]. S. Feghhi. (2015). Timing Only Traffic Analysis Project: Codes and Measurements. [Online]. Available: https://www.scss.tcd.ie/~feghhis/ ta_project/

[5]. X. Gong, N. Borisov, N. Kiyavash, and N. Schear, "Fingerprinting websites using remote traffic analysis," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2010, pp. 684–686.

[6]. D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multino-mial Naïve– Bayes classifier," in *Proc. ACM Workshop Cloud Comput. Secur. (CCSW)*, New York, NY, USA, 2009, pp. 31–42.

[7]. A. Hintz, "Fingerprinting websites using traffic analysis," in *Pri-vacy Enhancing Technologies* (Lecture Notes in Computer Science), vol. 2482, R. Dingledine and P. Syverson, Eds. Berlin, Germany: Springer, 2003, pp. 171–178.
[8]. M. Jaber, R. G. Cascella, and C. Barakat, "Can we trust the inter-packet time for traffic classification?" in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.
[9]. E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. SIAM Int. Conf. Data Mining*, 2001, pp. 1–11.
[10]. M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proc. 13th CM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2006, pp. 255–263.
[11]. L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *Computer Security* (Lecture Notes in Computer Science), vol. 6345, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds. Heidelberg, Germany: Springer, 2010, 199–214.
[12]. X. Luo *et al.*, "HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows," in *Proc. Netw. Distrib. Syst. Symp. (NDSS)*, 2011, pp. 1– 20.
[13]. B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, "I know why you went to the clinic: Risks and realization of HTTPS traffic analysis," in *Privacy Enhancing Technologies* (Lecture Notes in Computer Science), vol. 8555, E. De Cristofaro and S. J. Murdoch, Eds. Springer, 2014, 143–163.
[14]. A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website finger-printing in onion routing based anonymization networks," in *Proc. 10th Annu. ACM Workshop Privacy Electron. Soc. (WPES)*, New York, NY, USA, 2011, pp. 103–114.
[15]. 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 Symp. Secur. Privacy*, 2002, pp. 19–30. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1004359&newsearch=true&queryText=Statistical%20identification%20of% 20encrypted%20Web%20browsing%20traffic
[16]. T. Wang, X. Cai, R. Nithyanand, R. Johnson, and T. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proc. 23rd USENIX Secur. Symp. (USENIX Security)*, San Diego, CA, USA, Aug. 2014, pp. 143–157.
[17]. X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 116–130.
[18]. C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis," in *Proc. IEEE 16th Netw. Distrib. Secur. Symp.*, Feb. 2010, pp. 237–250. [Online]. Available: http://www.internetsociety.org/doc/traffic-morphingefficient-defense-against-statistical-trafficanalysis