A Successful Web Traffic Analysis Attack Using Only Timing Information

Saman Feghhi, Douglas J. Leith
Hamilton Institute, NUI Maynooth

***Abstract*—We introduce an attack against encrypted web traffic that makes use only of packet timing information on the uplink. This attack is therefore impervious to existing packet padding defences. In addition, unlike existing approaches this timing-only attack does not require knowledge of the start/end of web fetches and so is effective against traffic streams. We demonstrate the effectiveness of the attack against both wired and wireless traffic, achieving mean success rates in excess of 90%. In addition to being of interest in its own right, this timing-only attack serves to highlight deficiencies in existing defences and so to areas where it would be beneficial for VPN designers to focus further attention.**

I. Introduction

In this paper we consider an attacker of the type illustrated in Figure 1. The attacker can detect the time when packets traverse the encrypted tunnel in the uplink direction, but has no other information about the clients activity. The attacker's objective is to use this information to guess, with high probability of success, the web sites which the client visits. What is distinctive about the attack considered here is that attacker relies solely on packet timestamp information where the previously reported attacks against encrypted web traffic have mainly made use of observations of packet size and/or packet count information.

Our interest in timing-only attacks is twofold. Firstly, packet padding is a relatively straightforward defence against attacks that rely primarily on packet size, and indeed is currently either already available or being implemented in a number of popular VPNs [2]. Secondly, alternative attacks based on packet counting [2], [3] are insensitive to packet padding defences but require partitioning of a packet stream into individual web fetches in order for the number of packets associated with each web fetch to be determined, which may be highly challenging in practice on links where there are no clear pauses between web fetches. In contrast, packet timing-based attacks are not only largely unaffected by packet padding defences but also, as we will show, do not require partitioning of the packet stream. Hence, they are potentially a practically important class of attack against current and future VPNs. While some work has been carried out using inter-arrival time information to classify the application (HTTP, IMAP *etc*) [7], to our knowledge, there is no previous work reporting use of timing information alone to construct a successful attack against encrypted web traffic.

This work was supported by Science Foundation Ireland under Grant No. 11/PI/1177.

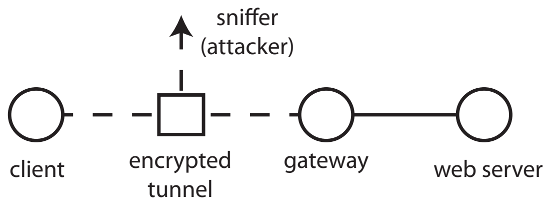


Fig. 1: Schematic illustrating attacker of the type considered. A client machine is connected to an external network via an encrypted tunnel (ssh, SSL, IPsec *etc*). The attacker can detect the time when packets traverse the tunnel in the uplink direction, but has no other information about the clients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

The main contributions of the present paper are as follows: (i) we describe an attack against encrypted web traffic that uses packet timing information alone, (ii) we demonstrate that this attack is highly effective against both wired and wireless traffic, achieving mean success rates in excess of 90% over ethernet and wireless tunnels and a success rate of 68% against Tor traffic, (iii) we also demonstrate that the attack is effective against traffic streams *i.e.* back to back web page fetches where the packet boundaries between fetches are unknown.

In addition to being of interest in its own right, particularly in view of the powerful nature of the attack, this timing-only attack also serves to highlight deficiencies in existing defences and so to areas where it would be beneficial for VPN designers to focus further attention. We note that, complementary to the present work, in [3] it is demonstrated that when the web fetch boundaries within a packet stream are known then an NGRAM approach using packet count together with uplink/downlink direction information is also sufficient to construct an effective attack against encrypted web traffic despite packet padding. Hence, we can conclude that (i) uplink/downlink packet ordering plus web fetch boundaries and (ii) uplink/downlink packet timing information are both sensitive quantities that ought to be protected by a secure encrypted tunnel. Packet padding does not protect these quantities. Directing defences against these two sets of packet stream features therefore seems an important direction for future work.

A. Related Work

The general topic of traffic analysis has been the subject of much interest, and a large body of literature exists. Some of the earliest work specifically focussed on attacks and defences for encrypted web traffic appears to be that of Hintz [6], which considers the SafeWeb encrypting proxy. In this setup (i) web page fetches occur sequentially with the start and end of each web page fetch known, and for each packet (ii) the client-side port number, (iii) the direction (incoming/outgoing) and (iv) the size is observed. A web page signature is constructed consisting of the aggregate bytes received on each port (calculated by summing packet sizes), effectively corresponding to the number and size of each object within the web page. In [13] it is similarly assumed that the number and size of the objects in a web page can be observed and using this information a classification success rate of 75% is reported.

Subsequently, Bissias *et al* [1] considered an encrypted tunnel setup where (i) web page fetches occur sequentially with the start and end of each web page fetch known, and for each packet (ii) the size, (iii) the direction (incoming/outgoing) and (iv) the time (and so also the packet ordering) is observed. The sequence of packet inter-arrival times and packet sizes from a web page fetch is used to create a profile for each web page in a target set and the cross correlation between an observed traffic sequence and the stored profiles is then used as a measure of similarity. A classification accuracy of 23% is observed when using a set of 100 web pages, rising to 40% when restricted to a smaller set of web pages.

Most later work has adopted essentially the same model as [1], making use of packet direction and size information and assuming that the packet stream has already been partitioned into individual web page fetches. In [9], [5] Bayes classifiers based on the direction and size of packets are considered while in [12] an SVM classifier is proposed. In [10] classification based on direction and size of packets is studied using Levenshtein distance as the similarity metric, in [11] using a Gaussian Bag-of-Words approach and in [14] using KNN classification. In [2] using a SVM approach a classification accuracy of over 80% is reported for both SSH and Tor traffic and the defences considered were generally found to be ineffective. Similarly, [3] considers Bayes and SVM classifiers and finds that a range of proposed defences are ineffective. In [4] remote inference of packet sizes from queueing delay is studied.

II. Anatomy of a Web Page Fetch

When traffic is carried over an encrypted tunnel, such as a VPN, the packet source and destination addresses and ports and the packet payload are hidden. We also assume here that the tunnel pads the packets to be of equal size, so that packet size information is also concealed, and that the start and end of an individual web fetch may also be concealed *e.g.* when the web fetch is embedded in a larger traffic stream. An attacker sniffing traffic on the encrypted tunnel is therefore able only to observe the direction and timing of packets through the tunnel, *i.e.* to observe a sequence of pairs $\{(t_k, d_k)\}$, $k = 1, 2, \cdots$ where t_k is the time at which the k'th packet is observed and $d_k \in \{-1, 1\}$ indicates whether the packet is travelling in the uplink or downlink direction. Since it will provide sufficient to mount an effective attack, we will assume a weaker attacker that can only observe the timestamps $\{t_k\}$, $k \in K_{up} := \{\kappa \in \{1, 2, \cdots\} : d_\kappa = -1\}$ associated with uplink traffic .

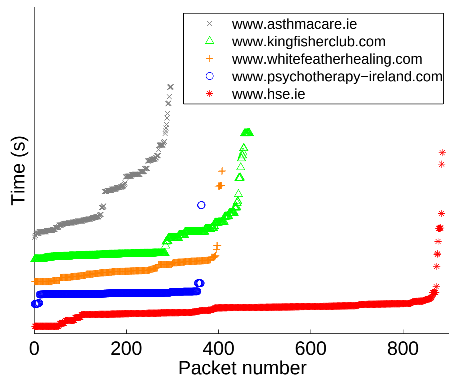


Fig. 2: Time traces of uplink traffic from 5 different Irish health-related web sites are shown. It can be seen that the web site time traces exhibit distinct patterns. The traces are shifted vertically to avoid overlap and facilitate comparison.

Figure 2 plots the timestamps $\{t_k\}$ of the uplink packets sent during the course of fetching five different health-related web pages (see below for details of the measurement setup). The x-axis indicates the packet number k within the stream and the y-axis the corresponding timestamp t_k in seconds. It can be seen that these timestamp traces are distinctly different for each web site, and it is this observation that motivates interest in whether timing analysis may by itself (without additional information such as packet size, uplink/downlink packet ordering *etc*) be sufficient to successfully de-anomymise encrypted web traffic.

To gain insight into the differences between the packet timestamp sequences in Figure 2 and, importantly, whether they are genuinely related to characteristics of each web page rather than to other factors, it is helpful to consider the process 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 issues an HTTP GET or POST request to which the server then replies. As the client parses the server response it issues 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.* when 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 each new server in order to issue the requests. Fetching of these objects may in turn trigger the fetching of further objects. Note that asynchronous fetching of dynamic content using, *e.g.* AJAX, can lead to a complex sequence of server requests and responses even after the page has been rendered by the browser. Also, typically the TCP connections to the various servers are held open until the page is fully loaded so that they can be reused for later requests (request pipelining in this way is almost universally used by modern browsers).

This web fetch process is illustrated schematically in Fig 3. We make the following more detailed observations:

1) *Connection to third-party servers*. Fetching an object located on a third-party server requires the opening of a new TCP connection to that server, over which the HTTP request is then sent. The TCP connection handshake

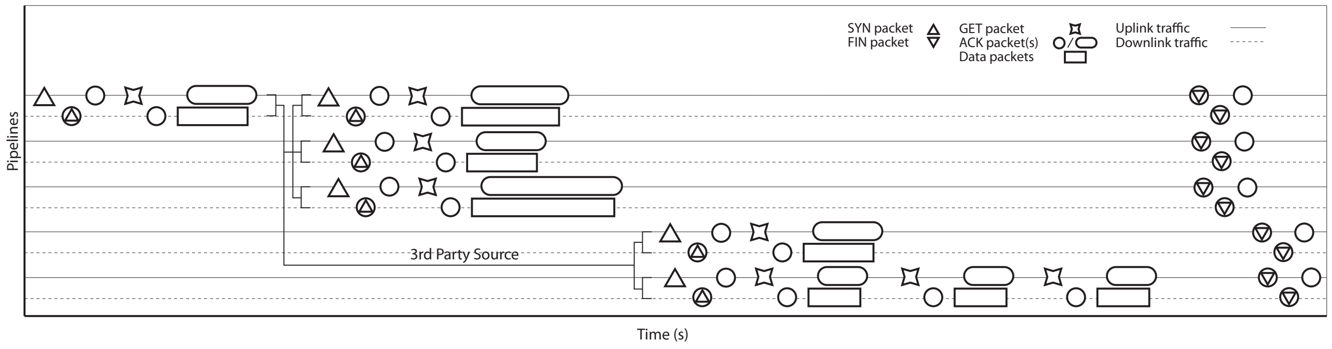


Fig. 3: This figure represent a typical web site query. It starts by requesting the index page. Then as the browser parses through this page more objects are fetched in parallel. Some objects may also be outsourced to 3rd party web sites which have their own pipelines. Dynamic content may be updated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

introduces a delay (of at least one RTT) and since the pattern of these 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 several GET/POST requests being sent to that server, one after another. Due to the dynamics of TCP congestion control, this burst of back-to-back requests can affect the timing of the response packets in a predictable manner that once again can potentially assist in identifying the web page.
3) *Asynchronous requests*. Dynamic content, *e.g.* pre-fetching via AJAX, can lead to update requests to a server with large inter-arrival times that can potentially act as a web page signature.
4) *Connection closing*. When 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 which allows it to be identified. 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 such as these, which vary depending upon the web page fetched, to create a timing signature which allows us to identify which web page is being fetched based on timing data only.

III. Comparing Sequences of Packet Timestamps

Suppose we have two sequences of packet timestamps $\boldsymbol{t} := \{t_i\}$, $i = 1, 2, \cdots, n$ and $\boldsymbol{t}' := \{t'_j\}$, $j = 1, 2, \cdots, m$. Note that for simplicity we re-label the uplink packet indices to start from 1 and to increase consecutively since none of our analysis will depend on this. Note also that the sequence lengths n and m are *not* assumed to be the same. To proceed we need to define an appropriate measure of the distance between such sequences.

A. Network Distortion of Timestamp Sequences

The packet stream observed during a web page fetch is affected by network events during the fetch. Changes in download rate (*e.g.* due to flows starting/finishing within the network) tend to stretch/compress the times between packets. Queueing within the network also affects packet timing, with queued packets experiencing both greater delay and tending to be more bunched together. Link-layer retransmission on wireless links has a similar effect to queueing. Similarly to changes in download rate, the effect is primarily to stretch/compress the times between packets.

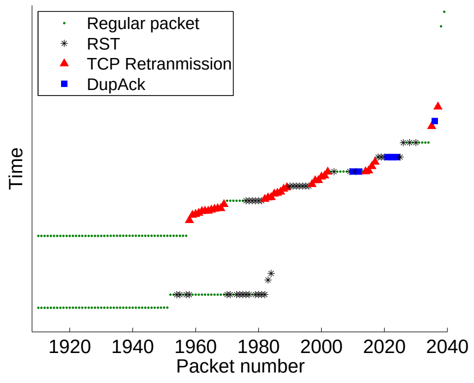


Fig. 4: Illustrating impact of changes in packet loss on the packet timestamp sequence. The bottom sequence shows the packet sequence at connection closing of a loss-free web fetch, while the top sequence shows the corresponding section from a different fetch of the same web page that was subject to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

Packet loss introduces a “hole” in the packet stream where the packet ought 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 example, Figure 4 shows uplink measurements of packet retransmissions and duplicate ACKs at the end of two fetches of the same web page where it can be seen that these have the effect of stretching the packet sequence.

B. Derivative Dynamic Time Warping

Our interest is in a measure of the distance between packet sequences which is insensitive to the types of distortion introduced by the network, so that the distance between packet streams $\boldsymbol{t}$ and $\boldsymbol{t}'$ associated with fetches of the same web page at different times is measured as being small, and ideally the distance between fetches of different web pages is measured to be large. To this end we use a variant of Dynamic Time Warping (DTW) [8]. DTW aims to be insensitive to differences between sequences which are due to stretching/compressing of time and so can be expected to at least partly accommodate the effects of changes in download rate, queueing delay *etc*.

We define a warping path $\boldsymbol{p}$ to be a sequence of pairs, $\{(p_k^i, p_k^j)\}$, $k = 1, 2, \cdots, l$ with $(p_k^i, p_k^j) \in V := \{1, \cdots, n\} \times \{1, \cdots, m\}$ satisfying boundary conditions $p_1^i = 1 = p_1^j$, $p_l^i = n$, $p_l^j = m$ and step-wise constraints $(p_{k+1}^i, p_{k+1}^j) \in V_{p_k^i, p_k^j} := \{(u, v) : u \in \{p_k^i, p_k^i + 1\} \cap \{1, \ldots, n\}, v \in \{p_k^j, p_k^j + 1\} \cap \{1, \ldots, n\}\}$, $k = 1, \cdots, l-1$. That is, a warping path maps points from one timestamp sequence to another such that the start and end points of the sequences match (due to the boundary conditions) and the points are monotonically increasing (due to the step-wise constraints). This is illustrated schematically in Figure 5, where the two timestamp sequences to be compared are indicated to the left and above the matrix and the bold line indicates an example warping path.

Let $P_{mn}^l \subset V^l$ denote the set of all warping paths of length l associated with two timestamp sequences of length n and m respectively, and let $C_{\boldsymbol{t},\boldsymbol{t}'}(\cdot) : P_{mn}^l \to \mathbb{R}$ be a cost function so that $C_{\boldsymbol{t},\boldsymbol{t}'}(\boldsymbol{p})$ is the cost of warping path $\boldsymbol{p} \in P_{mn}^l$. Our interest is in the minimum cost warping path, $\boldsymbol{p}^*(\boldsymbol{t}, \boldsymbol{t}') \in \arg\min_{\boldsymbol{p} \in P_{mn}^l} C_{\boldsymbol{t},\boldsymbol{t}'}(\boldsymbol{p})$. In DTW the cost function has the separable form $C_{\boldsymbol{t},\boldsymbol{t}'}(\boldsymbol{p}) = \sum_{k=1}^l c_{\boldsymbol{t},\boldsymbol{t}'}(p_k^i, p_k^j)$ where $c_{\boldsymbol{t},\boldsymbol{t}'} : V \to \mathbb{R}$, in which case optimal path $\boldsymbol{p}^*(\boldsymbol{t}, \boldsymbol{t}')$ be efficiently found using the backward recursion,

$$(p_k^i, p_k^j) \in \arg \min_{(p^i, p^j) \in V_k} C_{k+1} + c_{\boldsymbol{t},\boldsymbol{t}'}(p^i, p^j) \tag{1}$$

$$C_k = C_{k+1} + c_{\boldsymbol{t},\boldsymbol{t}'}(p_k^i, p_k^j) \tag{2}$$

where $V_k = (p^i, p^j) \in \{(u, v) : (p_{k+1}^i, p_{k+1}^j) \in V_{u,v}\}$, $k = l-1, l-2, \cdots$ and initial condition $C_l = c_{\boldsymbol{t},\boldsymbol{t}'}(n, m)$. When there is more than one optimal solution at step (1), we select (p_k^i, p_k^j) uniformly at random from amongst them.

A common choice of element-wise cost is the Euclidean norm $c_{\boldsymbol{t},\boldsymbol{t}'}(p^i, p^j) = (t_{p^i} - t'_{p^j})^2$. However, to improve robustness to noise on the timestamp values (in addition to misalignment of their indices), following [8] we instead use the following element-wise cost

$$c_{\boldsymbol{t},\boldsymbol{t}'}(p^i, p^j) = (D_{\boldsymbol{t}}(p^i) - D_{\boldsymbol{t}'}(p^j))^2 \tag{3}$$

where $D_{\boldsymbol{t}}(i) = \frac{(t_i - t_{i^-}) + (t_{i^+} - t_{i^-})}{2}$, $i^- = \max\{i-1, 1\}$ and $i^+ = \min\{i+1, |\boldsymbol{t}|\}$. Observe that $D_{\boldsymbol{t}}(i)$ is akin to the derivative of sequence $\boldsymbol{t}$ 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 $C(\boldsymbol{p}) = +\infty$ for paths $\boldsymbol{p} \in P_{mn}^l$ for which $|p_k^i - p_k^j| > \max\{w \min\{n, m\}, |m-n|\}$ for any $k \in \{1, \cdots, l\}$.

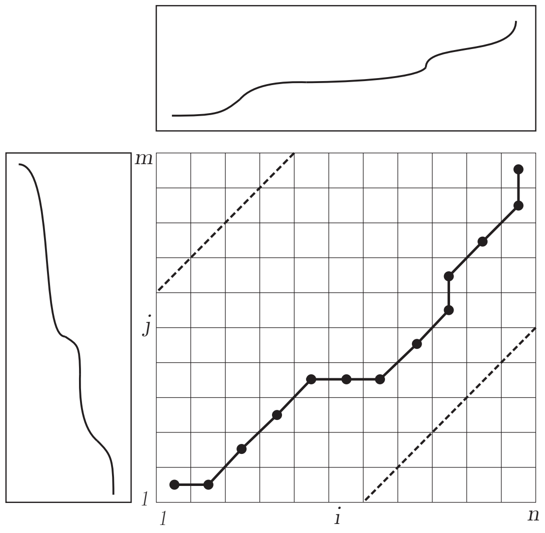
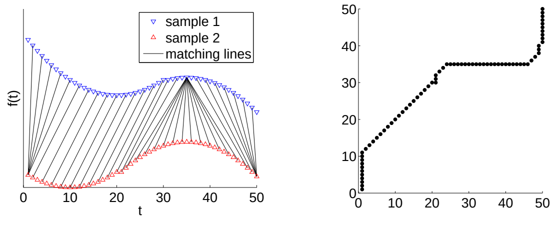
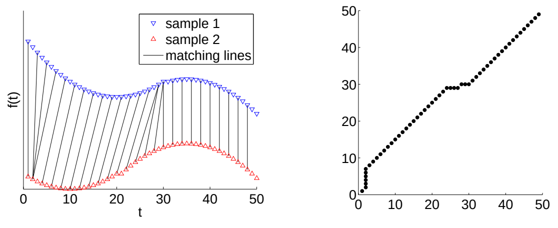


Fig. 5: Illustrating a warping path. The dashed lines indicate the warping window.



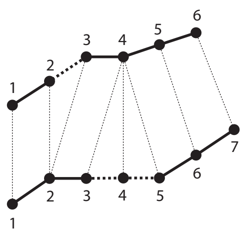
(a) Euclidean cost



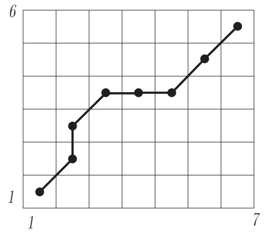
(b) Derivative cost

Fig. 6: Example DTW alignment and warping paths between two sequences vs cost function $c_{t,t'}$ 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.

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 visualisation 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 [8] this is known to be a feature of Euclidean cost. In comparison, use of the derivative distance tends to mitigate this effect and select a warping path with fewer horizontal and vertical sections.



(a) Sequence alignment.



(b) Warping path

Fig. 7: Illustrating method for calculating the F-distance between two timestamp sequences.

C. F-Distance Measure

Given two timestamp sequences, the warping path is a mapping between them. With reference to Figure 5, sections of the warping path which 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 which 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 refer to as the F-distance.

In more detail, let $\boldsymbol{p} = \{(p^i_k, p^j_k)\}$, $k = 1, \cdots, l$ be a derivative DTW warping path relating timestamp sequences $\boldsymbol{t}$ and $\boldsymbol{t}'$, obtained as described in the previous section. We partition the warping path into a sequence of subpaths within each of which either p^i_k or p^j_k remain constant and we count the subpaths which are longer than one. For example, for the setup shown in Figure 7 there are five subpaths: $(1, 1)$; $(2, 2), (2, 3)$; $(3, 4), (4, 4), (5, 4)$; $(6, 5)$; $(7, 6)$. Two of these subpaths consist of 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 $\kappa_1 := 0 < \kappa_2 < \cdots < \kappa_{r-1} < \kappa_r := l$ such that for each $s = 1, \cdots, r-1$ (i) either $p^i_{k_1} = p^i_{k_2}$ $\forall k_1,\ k_2 \in \{\kappa_s + 1, \cdots, \kappa_{s+1}\}$ or $p^j_{k_1} = p^j_{k_2}$ $\forall k_1,\ k_2 \in \{\kappa_s + 1, \cdots, \kappa_{s+1}\}$ and (ii) either $\kappa_{s+1} = l$ or condition (i) is violated for some $k_1,\ k_2 \in \{\kappa_s, \cdots, \kappa_{s+1} + 1\}$ *i.e.* each subsequence is maximal. Note that $p^i_k \neq p^j_k$ for all $k = 1, \cdots, l$ (due to warping path step-wise constraints) and so in condition (i) it is not possible for both p^i_k and p^j_k to be constant. We are now is a position to define the F-distance measure between timestamp sequences $\boldsymbol{t}$ and $\boldsymbol{t}'$, namely:

$$\phi(\boldsymbol{t}, \boldsymbol{t}') := \frac{\sum_{s=1}^{r-1} (\kappa_{s+1} - (\kappa_s + 1)}{n + m} \tag{4}$$

where κ_s, $s = 1, \cdots, r$ are the constant subsequences in minimal warping path $\boldsymbol{p}^*(\boldsymbol{t}, \boldsymbol{t}')$. It can be seen that $\phi(\boldsymbol{p})$ takes values in interval $[0, 1]$, and is 0 when sequences $\boldsymbol{t}$ and $\boldsymbol{t}'$ are identical (in which case the warping path $\boldsymbol{p}$ lies on the diagonal in Figure 5). For the example in Figure 7 the F-distance $\phi(\boldsymbol{p})$ is $(2 + 3)/13 = 0.385$.

IV. De-anonymising Web Fetches Over an Ethernet Tunnel

In this section we 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 data. The first dataset consists of 100 fetches of the home pages of each of the top 20 Irish health and the top 20 Irish finance web sites as ranked by www.alexa.com under its Regional/Europe/Ireland category in October 2013, yielding a total of 4000 individual web page fetches. In this dataset the browser cache is flushed between each fetch so that the browser always starts in a fresh state. In addition, a second dataset was collected consisting of the same 4000 web fetches but now without flushing of the browser cache between fetches. The web pages were fetched over a period spanning October 2013 to May 2014. A `watir-webdriver` script on Firefox 21.0 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 only packet timestamps on the uplink were actually used.

A. Hardware/Software Setup

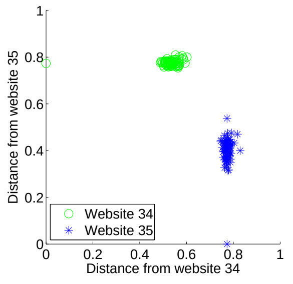
The network setup consists of a client that routes traffic via a tunnel to the gateway server. The gateway server forwards this client traffic to the Internet and routes responses back via the tunnel to the client. The tunnel is carried over a gigabit ethernet LAN and the gateway server also has a gigabit connection to the Internet. The client machine is a 3.00 GHz Core 2 Duo CPU with 2GB of RAM and the gateway server a 2.66 GHz Xeon CPU with 1GB of RAM Both machines are running Ubuntu Linux 12.04 LTS precise. The tunnel was implemented using custom software and netlink.

B. Classifying Measured Timestamp Sequences

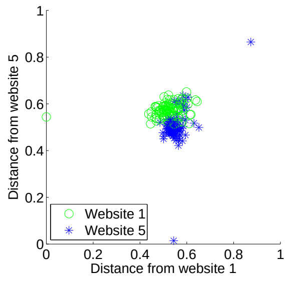
We use the F-distance measure $\phi(\cdot, \cdot)$ described in Section III to compare measured uplink timestamp sequences, with windowing parameter $w = 0.1$ unless otherwise stated.

Figure 8 shows example scatter plots obtained using this distance measure. In more detail, from the set T_i of measured timestamp sequences for the i'th web site we select a sequence $\boldsymbol{t}_i$ which minimises $\sum_{\boldsymbol{t} \in T_i} \phi(\boldsymbol{t}, \boldsymbol{t}_i)$ and then use $\boldsymbol{t}_i$ as the exemplar for the i'th web page. In Figure 8 we then plot $\phi(\boldsymbol{t}, \boldsymbol{t}_i)$ for each of the timestamp sequences $\boldsymbol{t}$ measured for web page i and also for timestamp sequences measured for another web page. In the example in Figure 8a it can be seen that the distance measure is indeed effective at separating the measured timestamp sequences of the two web pages considered into distinct clusters, so 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 two web pages is less distinct and so classification can be expected to be less reliable. As we will see, examples of this latter sort turn out to be fairly rare.

We considered two approaches for using $\phi(\cdot, \cdot)$ to classify timestamp sequences: K-Nearest Neighbours and Naive Bayes Classification.



(a)



(b)

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.

1) K-Nearest Neighbours: In this method, for each web page i we sort the measured timestamp sequences $\boldsymbol{t'} \in T_i$ used for training in ascending order of sum-distance $\sum_{\boldsymbol{t} \in T_i} \phi(\boldsymbol{t}, \boldsymbol{t'})$ and select the top 5 to use as exemplars to represent this web page. When 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 top K matches.

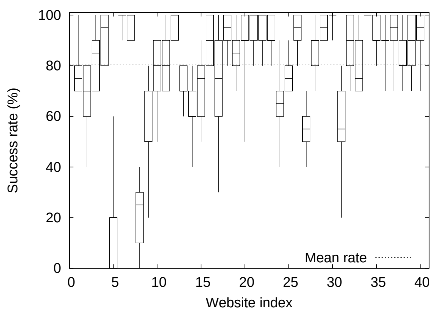
2) Naive Bayes Classifier: For each web page i from the measured timestamp sequences T_i used for training we select $\boldsymbol{t}_i \in \arg\min_{\boldsymbol{t'} \in T_i} \sum_{\boldsymbol{t} \in T_i} \phi(\boldsymbol{t}, \boldsymbol{t'})$ (in addition we also consider selecting $\boldsymbol{t}_i$ to minimise the variance of the distance ϕ, see below) and then fit a Beta distribution to the empirical distribution of $\phi(\boldsymbol{t}, \boldsymbol{t}_i)$ for $\boldsymbol{t} \in T_i$. Let $p_i(\cdot)$ denote the probability distribution obtained in this way. When presented with a new timestamp sequence $\boldsymbol{t}$, we calculate the probability $p_i(\boldsymbol{t})$ of this sequence belonging to web page i and select the web page for which this probability is greatest.

C. Experimental Results

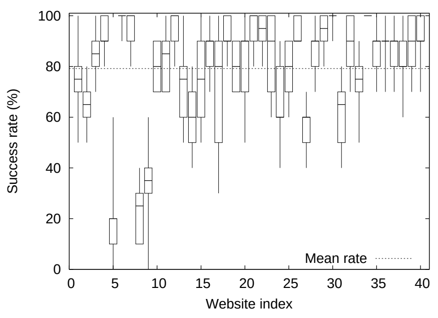
We begin by presenting results for the dataset where the browser cache is flushed between fetches. Figure 9 details the measured classification accuracy using the K-NN approach, for various values of K. We use 10-fold cross validation, where the 100 samples of each web site are divided into 10 random subsets and for each subset we use the remaining 90 samples as the training data to find the exemplars and use the 10 samples in the subset as the validation data. The rates for these 10 subsets for each web site are summarized and displayed in the figure. Each of the boxes indicate the 25%, 50% and 75% quartiles and the lines indicate the maximum and minimum values. The mean success rates for $K = 1$, $K = 3$ and $K = 5$ are 93.5%, 93.9% and 93.8% respectively.

Figure 10 plots the corresponding results obtained using the Naive Bayes approach. Performance is calculated when the exemplar for each web page is selected to minimise the mean and the variance of the distance. The mean success rates are 80.3% and 79.2% respectively.

Since the performance is consistently worse than that of the K-NN classifier we do not consider the Naive Bayes approach further in the rest of the paper.



(a) Minimum Mean



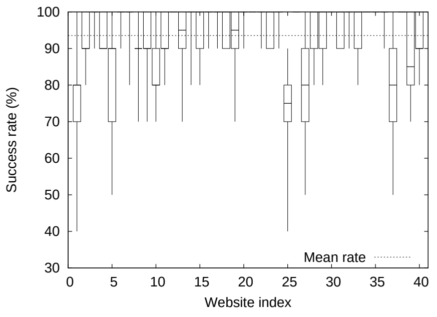
(b) Minimum Variance

Fig. 10: Naive Bayes classification performance, no browser caching.

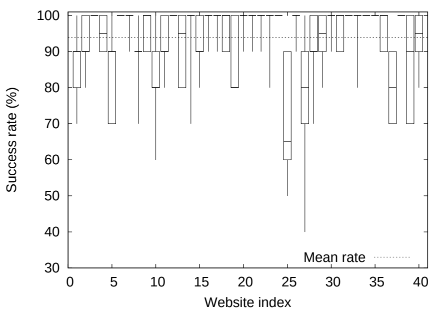
D. Standard vs. Cached: Different Versions of Same Web Page

On first visiting a new web page a browser requests all of the objects that form the web page. However, on subsequent visits many objects may be cached *e.g.* images, css and js files, *etc*. In the Mozilla browser, when the address of a web page is simply entered again shortly after the full page is fetched, since the cached copy of an object has not yet expired the cached copy will be used when rendering the web page and it will not be fetched over the network by the browser. But the browser can be forced to reload the web page by pressing F5 where it then sends a request for the objects and the server may either return an abbreviated NOT MODIFIED response if the cached object is in fact still fresh or return the full object if it has changed. Ultimately a full refresh can be induced by pressing Ctrl+F5 which requests for the full version of the web page as if no object is cached before. Hence, the network traffic generated by a visit to a web page may differ considerably depending on whether it has been visited recently (so the cache is fresh) or not.

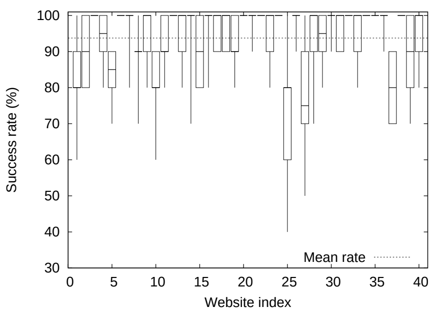
Classification of cached web pages can be expected to be more challenging than for non-cached pages since there is less network traffic and so less data upon which to base the classification decision. Figure 11 presents the measured classification accuracy when browser caching is enabled. We used the first version where the address is simply re-entered, to consider the worst case in terms of traffic length. It can be seen that, as expected, the classification accuracy is indeed reduced compared to Figure 9. However, the overall success rate for identifying web pages remains in excess of 85%.



(a) $K = 1$



(b) $K = 3$



(c) $K = 5$

Fig. 9: K-Nearest Neighbours classification performance, no browser caching.

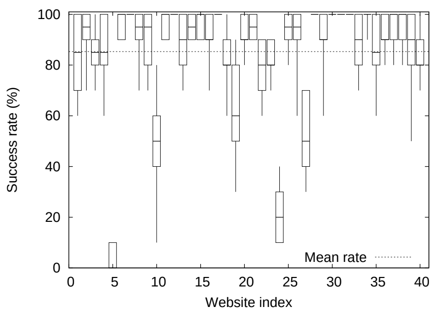


Fig. 11: K-Nearest Neighbour classification performance, with browser caching using 3 exemplars for each site. $K = 5$.

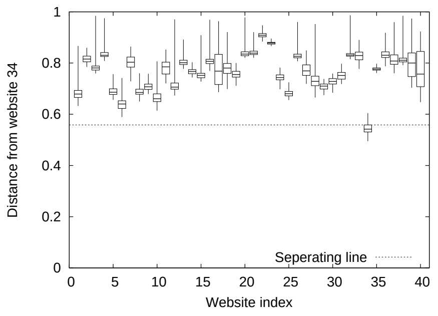


Fig. 12: Mean F-distance between the measured packet timestamp sequences in the training data set and the exemplar packet sequences for each web page, no browser caching.

E. False Positives

The experiments in the previous two sections are conducted with the assumption that the adversary knows that the web page that the user has visited is among the set of web pages for which training data has been collected.

When this assumption does not hold, *i.e.* the user might have fetched a web page outside of the adversary's training database, then we estimate the rate of false positives for K-Nearest Neighbour classification as follows. For each web page i in the training set we have 3 exemplar packet timestamp sequences as discussed in Section IV-B1. For every web page (including page i) in the training set, for each measured timestamp sequence we calculate the mean F-distance from the exemplar sequences for web page i. Figure 12 shows example measurements. Using this data we determine a threshold F-distance ϕ_i (indicated by the dashed line in Figure 12) and classify packet timestamp sequences for which the mean F-distance from the exemplars of web page i is greater than ϕ_i as not coming from web page i. In our study, we choose ϕ_i to be the 3rd quartile (75%) of the F-distance values of the target web page.

Using this classification approach Figure 13 plots the measured false positive rates. This data is obtained by removing the measurement data for each web page in turn from the training set, the measured packet timestamp sequences for this web page are then classified using this reduced training set and the false positive rate calculated as the fraction of the 100 packet timestamp sequences measured for each web page that are incorrectly classified as belonging to the reduced training set. It can be seen that the false positive rate is less than 2.5% for all web pages.

This maximum rate of 2.5% is lower than the complement of the success rate in Figure 9. This is to be expected as the classification task in Figure 13 is a coarser one. Namely, here we ask whether the measured packet sequence belongs to any web page within the training set or not, as opposed to the finer question in Figure 9 which asks to which specific web page within the training set does the packet sequence belong. That said, the false positive rate data in Figure 13 is necessarily an underestimate of the true false positive rate as it is determined from a small subset of all web pages on the Internet. However, this is unavoidable and we argue that the data is nevertheless sufficient to indicate that false positive rate is unlikely to be excessive in practice.

V. Measurement Results for Other Channels

In this section we extend consideration from ethernet to a number of different network channels. Namely, we consider 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. Similarly to before, in each case we collected packet timestamp data for 100 fetches of the home pages of each of the top 20 Irish health web sites as ranked by www.alexa.com.

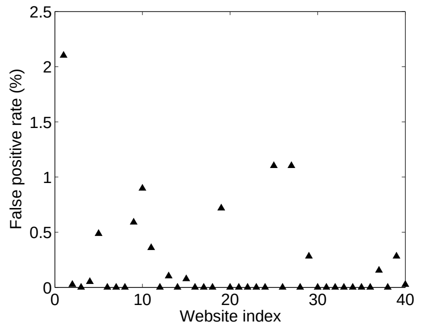


Fig. 13: Measured false positive rate, no browser caching.

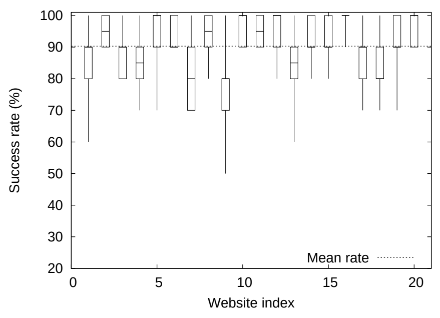
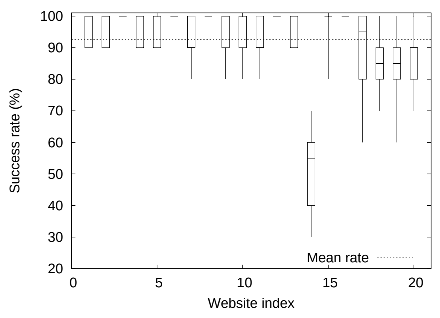
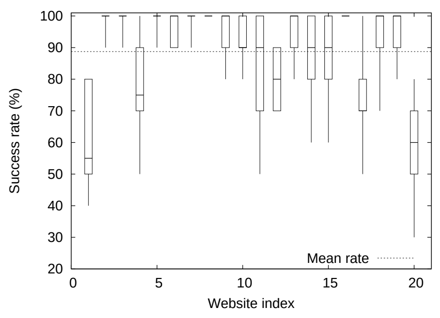


Fig. 14: Femtocell K-Nearest Neighbours classification performance, no browser caching, $K = 5$.



(a) Slot Time: 1ms



(b) Slot Time: 10ms

Fig. 15: Time-slotted tunnel K-Nearest Neighbours classification performance, no browser caching, $K = 5$.

A. *Femtocell Traffic*

A Femtocell is an eNodeB cellular base station with a small physical footprint (similar to a WiFi access point) and limited cell size (typically about 30m radius). It is intended to improve cellular coverage indoors, filling in coverage holes and improving download rates, while also offloading traffic from the macrocell 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 is a simple matter to route backhaul traffic via a sniffer. Mobile operators are, of course, aware of this and backhaul traffic is therefore secured via use of an IPsec encrypted tunnel. In the setup considered here, the Femtocell backhaul is over a university gigabit ethernet connection and we used *tcpdump* to log packets passing over this link.

1) Hardware/Software Setup: The client computer is a 3.00 GHz Core 2 Duo CPU with 2GB of RAM. It uses a Huawei K3770 HSPA USB Broadband Dongle to connect wirelessly to the internet via the Femotcell. The Femtocell is a commercial Alcatel-Lucent 9361 Home Cell V2-V device. The Femtocell wired backhaul is connected to the campus network via a NetGear EN 108 TP Ethernet hub. A monitor computer is also connected to this hub and logs all packets. The client and monitor computers both run Ubuntu Linux 12.04 LTS precise.

2) Results: In contrast to the relatively clean ethernet channel considered in Section IV-C, we found that traffic passing over the wireless femtocell link is often distorted by factors such as wireless and cellular noise, encoding/decoding delays, cellular control plane traffic *etc*. These distortions typically appear as shifts along the x-axis of the packet timestamp patterns and/or as delays in the y-axis. To compensate we increased the dynamic time warping window from 0.1 to 0.2. The measured performance using a K-NN classifier using 3 exemplars for each site and $K = 5$ is shown in Figure 14. The mean success rate is 90.3%, which compares with the mean success rate of 92.7% observed in Section IV-C when using a clean ethernet channel. It can be seen that use of the wireless channel tends to reduce 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 `iptables`, `netfilter` and `netfilter-queue`. The tunnel transports packets over a UDP channel in a time slotted fashion and the slot size is a configurable parameter.

1) Hardware/Software Setup: The experimenal setup is identical to that used in Section IV apart from the use of a customised tunnel. On the client computer all web traffic is captured using the OUTPUT `netfilter` hook, encapsulated into UDP packets and sent to a server at the other side of the tunnel. The server 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 which captures them using the PREROUTING hook, extracts the payload and forwards this to the application layer.

2) Results: Figure 15 shows the measured performance using a K-NN classifier where 3 exemplars are chosen from each site and $K = 5$. The overall success rate is 92.5% when the tunnel slot size is 1ms and 88.8% when the tunnel slot size is increased to 10ms. We also considered slot sizes larger than 10ms, but since we found such that large slot sizes tended adversely affect browser performance (and so would likely be problematic in practice) we do not include them here. This performance compares with a success rate of 92.7% over a plain ethernet tunnel. As might be expected, time-slotting decreases the classification success rate since it adds timing "noise". However, even with a relatively large slot size of 10ms the impact on performance is minor and so appears to be largely ineffective as a defence against our timing-based attack. Of course more sophisticated types of defence may be more effective, but we leave consideration of those to future work as they likely involve complex trade-offs between network performance and resistance to attack that we lack space to address here.

C. Tor Network

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

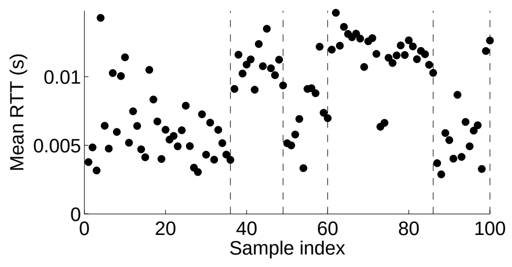
1) Hardware/Software Setup: The experimental setup is the same as in Section IV except that the traffic from the client browser, Mozilla Firefox 21.0, is proxified over Tor v0.2.4.23. As before, the browser cache is flushed between fetches.

2) Randomised Routing: Tor uses randomised routing of traffic over its overlay network in an attempt to make linking of network activity between source and destination more difficult. It can be expected that rerouting will have a significant impact on the timestamp sequence measured during 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. Differences in loss rate, queueing delay *etc* along different routes are also likely to impact measured timestamp sequences.

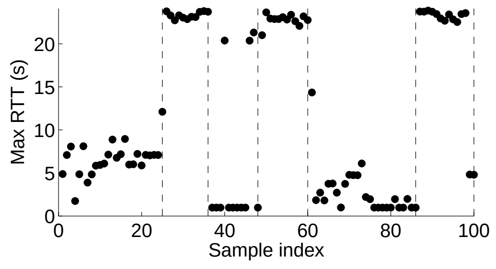
The impact of Tor rerouting on measured RTT is illustrated in Figure 16, which plots the mean and max delay between sending of a TCP data packet and receipt of the corresponding TCP ACK for repeated fetches of the same web page (although this information is not available to an attacker, in our tests it is of course available for validation purposes). Abrupt, substantial changes in the mean RTT are evident, especially in Figure 16b. These changes persist for a period of time as Tor only performs rerouting periodically.

Figure 17 illustrates the impact of Tor on the packet timestamps measured during a web page fetch.

3) Results: Figure 18 details the measured classification accuracy using the K-NN approach, where 3 exemplars are chosen from each site and a window size of $w = 0.2$ is used to accommodate the warping between samples. The mean success rate is 67.7% which compares with the mean



(a) Mean RTT for packets of each sample



(b) Max RTT for packets of each sample

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).

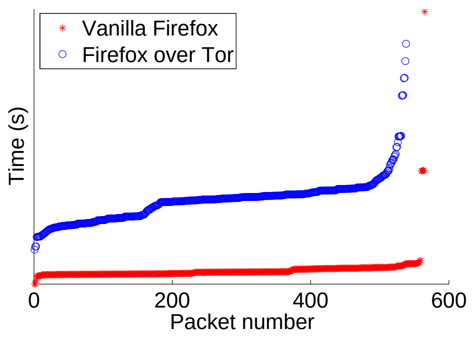


Fig. 17: Time traces of uplink traffic measured when fetching www.medicalcouncil.ie . Measurements are shown both when using vanilla Firefox and when using Firefox with the Tor plugin.

success rate of 93.0% when using a clean ethernet channel. As might be expected, use of the Tor network significantly reduces classification accuracy. However, the success rate of 67.7% compares with a baseline success rate if 5% for a random classifier over 20 web sites and so still is likely to represent a significant compromise in privacy. We note also that this compares favourable with the 54.6% rate reported by Panchenko *et al* in [12] against Tor traffic using packet size and direction information.

Channel		Number of Exemplars	w	Database size	K			
					1	3	5	7
Ethernet		5	0.1	40	93.52%	93.87%	93.75%	93.97%
		3	0.1	40	93.03%	92.33%	91.85%	-
		1	0.1	40	90.35%	-	-	-
		3	0.1	20	93.75%	93.15%	92.65%	-
Cached		3	0.1	40	86.53%	86.27%	85.33%	-
Slotted	1ms	3	0.1	20	94.15%	93.2%	92.55%	-
	10ms	3	0.1	20	90.25%	90.05%	88.75%	-
Femtocell		3	0.2	20	92 %	90.65 %	90.3 %	-
Tor		3	0.2	40	67.67 %	64.55 %	63 %	-
		3	0.2	20	68%	64.7%	63.3%	-
		3	0.1	20	68%	67.15%	64.55%	-

TABLE I: Summary of the measured success rate of the proposed attack reported here. Data is shown for different numbers of identifiers, different population sizes and different values of K in the K-nearest neighbours method.

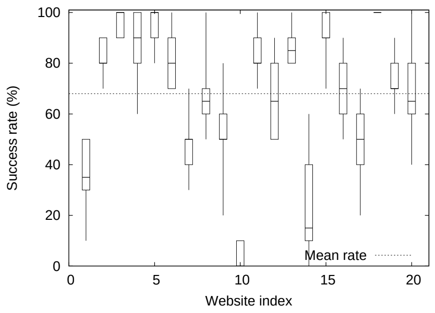


Fig. 18: Tor network K-Nearest Neighbours classification performance, no browser caching, $K = 1$.

VI. Finding a web page within a sequence of web requests

In the experiments presented so far we have assumed that within the observed packet timestamp stream the boundaries between different web fetches are known. This is probably a reasonable assumption on lightly loaded links where the link is frequently idle between web fetches. However, not only might this assumption be less appropriate on more heavily loaded links but it also allows for a relatively straightforward means of defence, namely insertion of dummy packets to obscure the boundaries between web fetches. In this section we therefore extend consideration to links where web fetches are carried out in a back to back fashion such that the boundaries between web fetches cannot be easily identified.

The basic idea is to sweep through a measured stream of packet timestamps trying to match sections of it against the timing signature of a web page of interest. This exploits the fact that our timing-only attack does not fundamentally depend on knowledge of the start/end times of the web fetch (unlike previous approaches which use packet counts to classify web pages).

In more detail, to locate a target web page within a stream of packet timestamps we first select three measured packet timestamp sequences for that web page to act as exemplars (as previously). Then, we sweep through the stream of timestamps in steps of 10 packets, extract a section of the stream of the same length as each exemplar and calculate the distance between the section and the exemplar. After sweeping through the full stream we select the location within the stream with least distance from the exemplars as the likely location of the target web page within the stream. While this process assumes that the target web page is present within the packet stream, using a similar approach to that in Section IV-E we could extend this approach to decide whether 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).

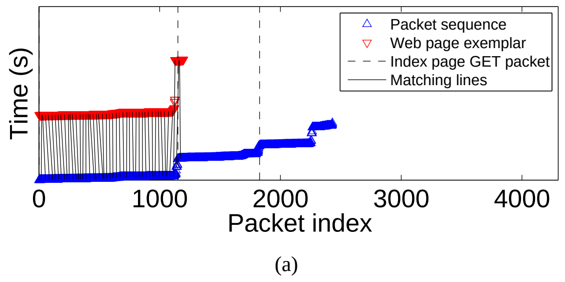
A. Results

We constructed a test data set as follows. We selected a subset of 5 web pages from those used previously. Namely: 1) medicalcouncil.ie 2) blueinsurance.ie 3) finfacts.ie 4) irish-health.com 5) firstireland.ie . We then selected 3 web pages uniformly at random from this set of 5, selected a permutation of these web pages uniformly at random and proceeded to fetch the pages in that order over an ethernet tunnel. We repeated this 100 times, so generating a dataset consisting of 100 measured packet timestamp streams.

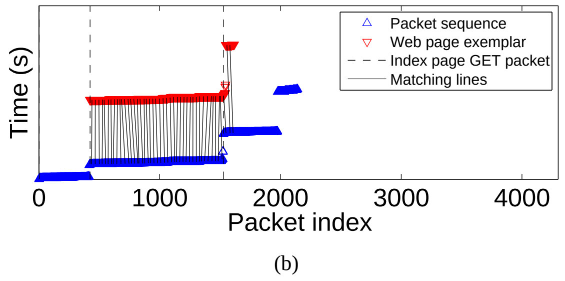
Using the classification approach described above we attempted to identify the location within each packet stream of one of the web pages (selected uniformly at random from the set of 3 web pages present in each measured packet stream). Figure 19 presents three examples of this, showing the position within a stream with least distance from the exemplars of a target web page. With this approach we achieved a success rate of 80% for locating the target web page within each packet stream to within a position error of ± 65 packets. Given the limited information being used, this is a remarkably high success rate and indicates the power of the timing-only attack. Further, of the 20% of cases where incorrect choices are made, in 7% the next best choice is the correct location of the target web page.

VII. Summary and Conclusions

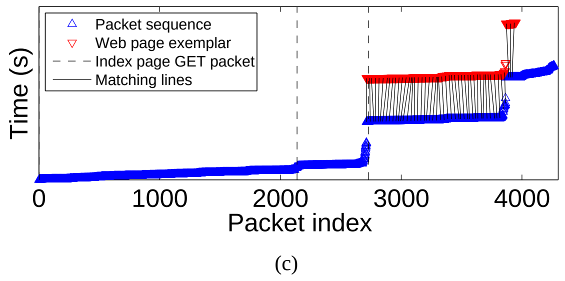
We introduce an attack against encrypted web traffic that makes use only of packet timing information on the uplink. In addition, unlike existing approaches this timing-only attack does not require knowledge of the start/end of web fetches and so is effective against traffic streams. We demonstrate the effectiveness of the attack against both wired and wireless traffic, consistently achieving mean success rates in excess of



(a)



(b)



(c)

Fig. 19: Illustrating locating of a web page within a packet stream. The target web site in our case study, shown with red triangles, is web site www.firstireland.ie. The vertical lines show the location of the first GET request for each web page. Following the order of the web sites in the text, the order of fetches in (a), (b) and (c) are $\{5, 4, 1\}$, $\{1, 5, 2\}$ and $\{3, 2, 5\}$ respectively.

90%. Table I summarises our measurements of the success rate of the attack over a range of network conditions.

Since this attack only makes use of packet timing information it is impervious to existing packet padding defences. We show that time slotting is also insufficient to prevent the attack from achieving a high success rate, even when relatively large time slots are used (which might be expected to significantly distort packet timing information). Similarly, randomised routing as used in Tor is also not effective. More sophisticated types of defence may be more effective, but we leave consideration of those to future work as they likely involve complex trade-offs between network performance (*e.g.* increased delay and/or reduced bandwidth) and resistance to attack that warrant more 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 would be beneficial 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 G Danezis and D Martin, editors, *Privacy Enhancing Technologies*, volume 3856 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg, 2006.

[2] X Cai, X.C. Zhang, Br 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*, CCS '12, pages 605–616, New York, NY, USA, 2012. ACM.

[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 *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 332–346, May 2012.

[4] X Gong, N Kiyavash, and N Borisov. Fingerprinting websites using remote traffic analysis. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 684–686, New York, NY, USA, 2010. ACM.

[5] 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*, CCSW '09, pages 31–42, New York, NY, USA, 2009. ACM.

[6] A Hintz. Fingerprinting websites using traffic analysis. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 171–178. Springer Berlin Heidelberg, 2003.

[7] M. Jaber, R.G. Cascella, and C. Barakat. Can we trust the inter-packet time for traffic classification? In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5, June 2011.

[8] E.J. Keogh and M.J. Pazzani. Derivative dynamic time warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11. 2001.

[9] M Liberatore and Brian N Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 255–263, New York, NY, USA, 2006. ACM.

[10] L Lu, E.C. Chang, and M.C. Chan. Website fingerprinting and identification using ordered feature sequences. In D Gritzalis, B Preneel, and M Theoharidou, editors, *Computer Security ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 199–214. Springer Berlin Heidelberg, 2010.

[11] 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. *ArXiv e-prints*, March 2014.

[12] A Panchenko, L Niessen, An 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*, WPES '11, pages 103–114, New York, NY, USA, 2011. ACM.

[13] Q. Sun, D.R. Simon, Yi-Min Wang, W. Russell, V.N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 19–30, 2002.

[14] W Tao, C Xiang, N Rishab, R Johnson, and I Goldberg. Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, San Diego, CA, August 2014. USENIX Association.