Distinguishability and Web Traffic Timing Analysis

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Saman Feghhi

School of Computer Science and Statistics
Trinity College Dublin

Supervisor: **Prof. Douglas J. Leith**

This dissertation is submitted for the degree of
Doctor of Philosophy

Trinity College Dublin April 2017

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and that this thesis has not been submitted as an exercise for a degree at this or any other university. This dissertation is entirely my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Saman Feghhi
April 2017

Abstract

Privacy of data transmitted over public networks has attracted much attention over recent years. Studies show that attacks based on traffic analysis enable malicious users to extract useful information about communications between parties, even if the content of these communications is carried over an encrypted channel.

In this thesis we begin by introducing a website fingerprinting attack against encrypted web traffic that uses only packet timing information on the uplink. This attack is therefore impervious to existing packet padding defence schemes. In addition, unlike existing methods this 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%. We also consider an attacker that can collect training data, but only over a different connection from that against which the attack is directed. This is a significantly easier to perform attack than one which depends on training data collected over the victim link. We demonstrate that an attacker can infer the correct web page >87% of the time when the training data is collected at a distance of up to 25 km from the victim, provided that the type of link is similar *e.g.* if the victim link uses a cable modem then the training data should be measured over a cable modem link. We also investigate the impact of distance in time between when the training data is collected and when the attack is performed.

We then move on to consider defences against timing analysis attacks. We initiate the study of the joint trade-off between privacy, throughput and delay in a shared network as a utility fairness problem and derive the proportional fair rate allocation for networks of flows subject to privacy constraints and delay deadlines. Since this analysis is confined to Bernoulli traffic arrivals we then consider the design of a timing analysis resistant encrypted tunnel that admits general traffic arrivals. The basic idea is to ensure privacy by serving the incoming traffic using predefined traffic patterns, called "traces". The service rate is controlled by activating sufficient number of traces to match the rate of arrivals. The delay, throughput and privacy performance achieved is evaluated using a prototype implementation of a privacy-enhanced VPN.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

One of the notable changes in the Internet in recent years from the point of view of the general public is probably the growth of ubiquitous surveillance of network traffic both by nation states and commercial entities such as ISPs. These are powerful adversaries. Currently the primary technique for protecting network traffic from snooping is by use of encryption, and use of encryption is rapidly growing e.g. with the increasingly widespread deployment of HTTPS. This is effective at concealing the contents of individual packets, but it is also known that the packet streams can still reveal information about user activity e.g. via the packet sizes, timing, source/destination IP address etc. Studying attacks and defences based around this sort of packet stream information is the focus of this thesis. Our aim is to gain an improved understanding of attacks that leads to the design of better defences.

It is the increasing volume of sensitive data transmitted over the Internet that is leading both to increasing surveillance and to increased concerns by users regarding their privacy. Third parties, ranging from an advertising company interested in users' tastes and preferences, to government bodies that attempt to monitor the browsing behavior of the population, seek to extract potentially sensitive information from the vast amount of data now available online. Apart from trusting content holders with sensitive information which, has developed a growing concern on its own [25, 63, 68], users are exposed to a range of so-called Man in the Middle (MitM) attacks where the attacker aims to eavesdrop or alter the messages transmitted between parties over a public link [9, 32, 56, 62, 97]. This promotes the use of privacy-enhancing technologies that allow users to embrace the advantages of online services while protecting their privacy when sharing their personal information online.

For many of reasons users prefer to conceal their browsing activity from potential adversaries. According to Tor [70] some of these reasons include:

- to protect against marketers or anyone else willing to pay for your internet browsing records. This information can be full record of every site you visit, the text of every search you perform and potentially userid and even password information of every login attempt on the internet.
- to protect against breaches and betrayals, from losing backup tapes to giving away the data to researchers.
- to protect research sensitive topics. For example accessing information on AIDS, birth control or world religion might be behind a national firewall in some countries.
- even harmless web browsing can sometimes raise red flags for suspicious observers. It is then important to protect your browsing behaviour against any sort of surveillance.

Therefore, a large body of research and development is dedicated to implementation of privacy-enhanced tunnels and VPNs to aid users with secure and anonymous browsing. One of the most renowned examples is the Tor Project which protects the identity of users by transmitting their traffic through a network of anonymous nodes. Other examples include Freegate, Ultrasurf, etc. which focus on anonymous browsing and are designed to address limitations of freedom-of-internet in certain countries.

Current privacy-enhancing technolgies use encryption, rerouting and IP obfuscation as primary approaches to remove linkability between users and their traffic. In this thesis we consider traffic analysis; a topic that is neglected in most of privacy-enhancing technologies. Numerous studies show that traffic analysis attacks enable an adversary to draw information about communications between parties even if the content of messages is hidden through the use of an encrypted channel [12, 65, 73, 96]. For example studies show that the service or application in use can be identified [1, 2, 49, 61, 64, 94] and in the case of web browsing that the particular web sites being browsed by the users can be readily identified [8, 34, 35, 60].

The aim of this thesis is to address the shortcomings of existing approaches and provide an ultimate solution for web privacy. We consider traffic analysis attacks and defences for web traffic, a subject that has become of increasing interest in recent years with the wider rollout of HTTPS and use of encrypted Virtual Private Networks (VPNs). The majority of literature in this area focuses on attacks that make use

of packet size, direction, ordering *etc.* [35, 59]. However these types of attacks can often be countered by means of inexpensive defences such as packet padding [40, 95]. Instead we consider the attacks using only timing information where the assumption is that, apart from the channel being encrypted, all packets are also padded to be of the same size and the beginning and end of web pages fetches is unknown. The only information available to the adversary then is the sequence of inter-arrival times between packets.

In addition to being of interest in its own right, such timing-only attacks serve to highlight deficiencies in existing defences and so to areas where it would be beneficial for VPN designers to focus further attention. Indeed our main interest in constructing attacks is to demonstrate the need for more effective defences and to assist with evaluating the effectiveness of such defences. We also aim to evaluate the overheads in terms of bandwidth and delay associated with such defences, given multiple users are often co-existing over the same links.

1.2 Contributions

The main contributions of this work are (i) introducing a timing-only traffic analysis attack which is resistant to inexpensive means of defence, and (ii) providing countermeasures to such attacks and addressing the trade-off between privacy, delay and throughput overhead through the design, analysis and implementation of a traffic analysis resistent VPN.

We begin by introducing an attack against encrypted web traffic that only uses 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 any knowledge of the start/end of web fetches and so is effective against traffic streams. We evaluate the robustness of the proposed attack over variety of channels and network conditions. We also consider an attacker that can collect training data, but only over a different connection from that against which the attack is directed. This is a significantly easier to perform attack than one which depends on training data collected over the victim link. We demonstrate that an attacker can infer the correct web page >87% of the time when the training data is collected at a distance of up to 25 km from the victim, provided that the type of link is similar *e.g.* if the victim link uses a cable modem then the training data should be measured over a cable modem link. We also investigate the impact of distance in time between when the training data is collected and when the attack is performed.

Following consideration of attacks we then move on to consider defences. We initiate the study of the joint trade-off between privacy, throughput and delay in a shared network as a utility fairness problem and derive the proportional fair rate allocation for networks of flows subject to privacy constraints and delay deadlines. Since this analysis is confined to Bernoulli traffic arrivals we then consider the design of a timing analysis resistant encrypted tunnel that admits general traffic arrivals. The basic idea is to ensure privacy by serving the incoming traffic using predefined traffic patterns, called "traces". The service rate is controlled by activating sufficient number of traces to match the rate of arrivals. The design aims to address the deficiencies of current privacy-enhancing technologies by providing a defence against all types of traffic analysis attacks. Unlike existing defenses in the literature, presented work investigates the tradeoff between privacy and delay and throughput overhead. A prototype of the presented defence also shows that it has a reasonable performance in terms of delay and throughput overhead making it a practical model to implement a traffic analysis resistant VPN.

1.3 Structure

The structure of this thesis is as follows

- In Chapter 2 we provide a brief overview of some of the concepts and technical terms that are used throughout this thesis.
- In Chapter 3 we review the current literature in traffic analysis to address the limitations of attacks and defenses that are proposed.
- In Chapter 4 we introduce a web traffic analysis attack that makes use of only timing information. The proposed model is evaluated over an ethernet channel.
- In Chapter 5 we evaluate the performance of our attack in a wide range if scenarios, including different versions of web pages fetched, different network types (femtocell, Tor *etc.*) and different link speeds, noises, web page length *etc.* Moreover, the robustness of the attack is evaluated for cases where the training data is collected on a link remote from the victim and at a time up to 16 weeks different from the user's queries.
- In Chapter 6 a fair defence against timing-only attacks is proposed. The traffic flow considered in this chapter consists of Bernoulli arrivals following an on-off pattern, where user transmission is divided into active and silent cycles.

- In Chapter 7 we consider the design of a timing analysis resistant encrypted tunnel that admits general traffic arrivals. The delay, throughput and privacy performance achieved is evaluated using and prototype implementation of a privacy-enhanced VPN.
- Concluding remarks are given in Chapter 8 along with a discussion on limitations and future work.

1.4 Publications

The following journal and conference papers have been published or are under submission during the course of this Ph.D..

1. Feghhi, S. and Leith, D. J. (2016b). A Web Traffic Analysis Attack Using Only Timing Information. In *IEEE Transactions on Information Forensics and Security, 11(8)*, pages 1747-1759. [22]
2. Feghhi, S., Leith D. J. (2016b). Privacy-Enhanced VPN Resistant to Timing-Analysis, submitted to *IEEE/ACM Transactions on Networking*.
3. Feghhi, S. and Leith, D. J. (2016a). A First-Hop Traffic Analysis Attack Against a Femtocell. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1060-1065. [20]
4. Feghhi, S., Leith D. J., Karzand, M. (2016b). Proportional Fair Rate Allocation for Private Shared Networks. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 1006-1011. [23]
5. Feghhi, S., Leith D. J. (2017a). Time and Place: Robustness of a Traffic Analysis Attack Against Web Traffic. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*. [21]

Chapter 2

Background

In this chapter we provide an overview of the preliminary knowledge of networking and machine learning that would help understanding the concepts that we present in this thesis. We begin by giving a brief explanation of some of the definitions in networking and network protocols followed by a detailed explanation of the threat model that is studied in this thesis. We also review some definitions in machine learning, statistics and classification techniques that are considered in this thesis.

2.1 Computer Networking

A computer network is a telecommunication network which allows nodes to share resources [80]. The nodes in this network are computer devices that are interconnected with data links. The link between network nodes, can be wired using technologies like coaxial cables, twisted pairs, optical fibres, etc. or it can be wireless like terrestrial microwave, satellite, cellular, etc. The information traversing over network links is split into small chunks of data messages called *packets*. A packet much like a regular mail consists of a header which resembles an envelope that contains addresses of parties involved in that message. The other part of a packet is payload which is like the actual letter inside the envelope. Transmission and routing of packets are regulated via network protocols.

2.1.1 Internet Protocol Suite

Internet protocol suite is a conceptual model with a set of protocols that describes the internet and similar networks [82]. It is developed by IETF prior to the OSI model, a more structured model that represents any form of communication network. It is

sometimes referred to as TCP/IP after two of its principal protocols. The Internet protocol suite provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed and received. This functionality is organized into 4 abstraction layers which are used to sort all related protocols according to the scope of networking involved. These layers are:

1. **Link Layer**: This is the lowest component of TCP/IP. It includes the protocols for node-to-node data transfer. The nodes in this layer are usually (but not necessarily) physically connected. The protocols in this layer are in charge of successful transmission of data on the physical layer with detection and possibly correction of errors. Some of the protocols in this layer include: MAC (Ethernet, DSL, ISDL, etc.), IEEE 802.11 (Wi-Fi), PPP (point-to-point protocol) and L2TP (a tunneling protocol).

2. **Network Layer**: This layer is in charge of transferring data sequences between two nodes that reside on the same network. The nodes in this layer are provided with addresses that are used by routers to deliver packets to their corresponding destinations. If messages in this layer are too large, the protocols are also in charge of splitting them into smaller messages and reassemble them at destination.

    - **Internet Protocol (IP)**, the principal communication protocol, is a member of this layer. It is solely responsible for delivering packets from host to destination using IP addresses [81]. It also defines the addressing methods over the Internet which is coded into IP headers that encapsulates the data.

3. **Transport Layer**: The protocols in this layer are responsible for transferring variable length data sequences from a source to a destination host via one or more networks while maintaining the quality of service function. The transport layer controls the reliability of a given link through flow control, segmentation/desegmentation and error control. TCP and UDP are well known protocols in this layer.

    - **TCP** provides a reliable, ordered, and error-free delivery of packets between applications communicating by an IP network [86]. It is also responsible for congestion avoidance and control through the transmission of acknowledgements. The acknowledgement packets (ACKs) in both directions are used to infer network conditions between senders and receivers.

- **UDP** is a simpler protocol of this layer. Unlike TCP, it does not require handshaking or prior communications to setup transmission channels. Also since there is no acknowledgement mechanism in this protocol, there is no guaranteed delivery, ordering or duplicate protection. UDP protocol is used in applications where reliable delivery is not required or it is handled in application layer [88].

4. **Application Layer** is the closest layer to the user. The protocols in this layer are used by most applications for providing user services or exchanging application data over the network connections established by the lower level protocols. It is a general layer to address last three layers of OSI model namely, Session, Presentation and Application layers. The application layer protocols often ignore the information about lower layers considering them as black boxes. Examples of application protocol include: HTTP, FTP, SMTP and DHCP. MIME and encryption protocols such as SSL and TLS which are presentation layer protocols are also members of TCP/IP's application layer.

2.1.2 Threat Model

Unless network nodes reside in a private network, regardless of using a wired or wireless connection, their communication is exposed to interception and eavesdropping. The chances are higher for wireless links e.g. Wi-Fi as it is easier for an adversary to eavesdrop the channel without being noticed or having to physically intercept the channel. An attack of this kind is known as a Man-in-the-Middle (MitM) attack. In this attack, an adversary secretly relays and possibly alters the communication between two parties who believe they are directly communicating between each other.

To protect data transmission against MitM attacks, public channels can be encrypted. The encryption can be applied to the payload to protect the content of messages, or to the entire message (including the headers) through protocols such as IPSec, which would further obfuscate addresses of participating parties. Depending on the purpose of the protection, different methods of encryption are considered:

- **Cryptographic Protocols**: This is a security measure carried out by means of internet protocols to protect individual services over unsecured public networks such as internet. Examples of it include Secure Shell (SSH), TLS (Secure HTTP, etc) and SSL (for VoIP communications, etc.)

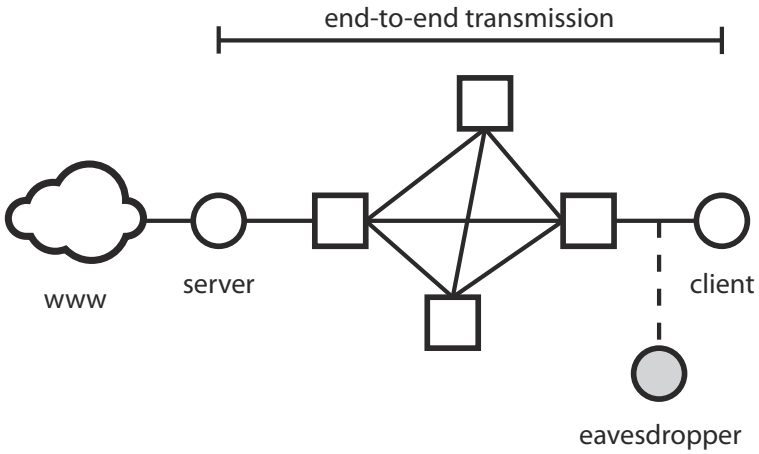


Fig. 2.1 Schematic illustrating the threat model.

- **Virtual Private Networks**: In this method communication between two private nodes are extended over the public network. This allows users residing in different local networks to securely communicate over internet as if they are directly connected to each other.

Figure 2.1 illustrates the thread model considered in this thesis. The attacker in this model is snooping the communication between clients and an end server which is responsible for secure content delivery. Although adversaries can intercept any of the links between two parties, we generally consider first-hop attacks where attackers directly tap the first link between user and the Internet. They can range from a malicious neighbour connecting to your router and monitoring your traffic or an adversary snooping your Wi-Fi or mobile phone communications.

The protection is not specific to a particular internet service e.g. browsing, or VoIP but to all communication over this link. Therefore, the aim is to provide an end-to-end protection against MitM attacks between clients and service providers while providing a reasonable quality of experience (QoE). This is done by creating a secure tunnel between clients and a server that is not only protected with standard methods of encryption, but also provides mitigation against traffic analysis attacks.

2.2 Machine Learning

Machine learning is a sub-field of computer science in artificial intelligence that studies the algorithms that can learn and make predictions about data without being explicitly programmed [84]. It is particularly used for cases where a clear model for data is absent

and therefore needs to be made from previous data samples. Some of the applications of machine learning are spam filtering, anomaly detection, search engines, computer vision and voice recognition. There are two main procedures in each machine learning techniques. First the method is trained with a large subset of data and a model is generated. Then this model is tested against a new subset of data to measure the performance of the algorithm against future unobserved samples.

2.2.1 Types of Tasks

Depending on the nature of learning parameters available to the system, machine learning tasks are classified in three main categories:

- **Supervised learning**: In this type of problems the computer is trained with a set of example inputs where the outcome is known. The algorithm learns the features from these examples and uses this knowledge to predict the labels for future unobserved instances. Decision trees, naive Bayes classifier, K-nearest neighbour are among supervised learning algorithms.
- **Unsupervised learning**: Unlike supervised learning the observed examples in this case are not labeled. Instead, the algorithm constructs a model to describe the hidden structures in training data [87]. Future examples are then tested against this model and prediction about whether they follow the same structures are made. Since the data is not labeled in this class of learning methods, the accuracy of predictions relies on the number of outliers in training data. Some examples of unsupervised methods include: K-means, anomaly detection and neural networks.
- **Reinforcement learning**: In this area, computer programs interact with a dynamic environment in which they must perform a certain goal. They would then learn to navigate in their problem spaces by monitoring the outcomes of different actions.

2.2.2 Approaches

Based on the size of data and nature of features, various machine learning algorithms have been introduced. We present a brief description of two of these algorithms that are used in this thesis.

- **K-nearest neighbour**: This is a supervised non-parametric algorithm used for both classification and regression [83]. For KNN regression, the output is the property value of the object calculated from the average of the values of its neighbours. However when it is used for classification (as it is in this thesis), the output is a class assignment based on a popular vote among the neighbours. K-NN is a lazy and easy to implement algorithm and has a better performance on small feature sets where the curse of dimensionality does not affect the accuracy of results.

  Methodology: Assume the goal is to classify samples into N categories. Let $\{x_1, ..., x_m\}$ be the feature set of samples. Moreover, let $R_{C_i} = \{r_1, ..., r_{|R_{C_i}|}\}$ be the exemplars representing class C_i. In order to predict class memberships, each sample is compared to all exemplars and the distance between the feature sets are calculated and the values are sorted in ascending or descending order. A sample then belongs to a class of which exemplars appear most among first K values of the ordered set.

- **Naive Bayes classifer**: This is a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features [85].

  Methodology: Let $\mathbf{x} = \{x_1, ..., x_n\}$ be the feature set of a problem instance that is being classified into K possible classes C_k. Following Bayes' theorem, the probability of x belonging to class C_k can be decomposed as

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \tag{2.1}$$

  where $p(C_k)$ is the prior, $p(x|C_k)$ is the likelihood and $p(x)$ is the evidence. Now assume the classes are equiprobable, hence priors = 1 / K. The probability of evidence is also equal for all instances. So probability of a sample being from class C_k can be directly calculated from the likelihood $p(x|C_k)$. This value can be computed by fitting a model to training samples of different classes. For example if we assume the samples are of normal distribution parameterized by μ_c and σ_c^2, the likelihood of sample x belonging to class c is

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}} \tag{2.2}$$

once all $p(C_k|x)$ are calculated sample x belongs to the class with minimum probability.

2.2.3 Error Rates

There are four different outcomes when making predictions about presence of a condition in unobserved data:

- **True positive**: the algorithm detects a condition when the condition is present.
- **True negative**: it does not detect the condition when the condition is absent.
- **False positive**: it detects a condition when it is absent. This is referred to type I error
- **False negative**: it does not detect a condition where it is present. This is referred to type II error.

The performance of a learning algorithm is measured by evaluating combinations of these rates e.g. accuracy, sensitivity and specificity and etc. A well trained algorithm should have minimum false positive and false negative rates.

2.2.4 Overfitting

When fitting a model to training data, two things can lead to increased error rates:

- **Overfitting**: The model can be excessively complex, with too many parameters. This model is often highly accurate in predicting the outcome of training data that is already known but it fails to make predictions for unobserved samples.
- **Underfitting**: The model is too simple to capture main trends in data. An example is fitting a linear model to a non-linear data.

2.2.5 Model Assessments

Classification algorithms can be validated by accuracy assessment techniques. Some of these techniques include:

- **Holdout method**: In this method the data split into training and test sets where the model from training is evaluated against the test set. Usually the split is 2/3 training set and 1/3 test set designation.

- **k-fold cross validation**: In this method the data is randomly split into k sets where $k-1$ sets are used for training and the trained model is evaluated against the k-th set. This procedure is repeated multiple times until the accuracy of the model is within a threshold.
- **Bootstrapping**: In this method the test set is selected by randomly selecting n samples with replacement. Similar to the other methods the procedure is repeated and each time trained model is evaluated against the test set.

For either of these methods, by calculating the error rates and other statistical measures of their family such as accuracy, sensitivity, specificity, ROC curve, etc., the parameters of learning algorithms are modified until the error rates are within an acceptable threshold.

Chapter 3

Literature Review

3.1 Network Privacy Attacks and Defences

The topic of secure data transmission has been studied in a great extent. Unlike the work of this thesis that considers packet stream information leak, a large body literature addresses cryptographic attacks and defences. One of the major threats in this area is Deep Packet Inspection (DPI). An example of such attack is discussed in [89]. The authors proposed `ScrambleSuit` as a defence against DPI attacks. Another extensive DPI attacks evaluation is carried out in [17] where a defence called format transforming encryption (FTE) is tested against different DPI attacks.

3.2 Traffic Analysis

The general topic of traffic analysis has been the subject of much interest, and a large body of literature exists [3, 27, 39, 76]. One of the earlierst work is that of Song *et al* [65], where the attacker can learn significant information about what users type in SSH sessions using packet count and timing patterns of key strokes. In [15], authors discuss an attack over non-HAS video streaming where the adversary is able guess titles of videos being watched by 95% accuracy using various learning methods. The population consists of 100 samples of 100 videos totalling 10000 traces. White et al. [79] were able to detect phonemes in encrypted VOIP sessions by fragmenting packet sequence into subsequences. They claim that packet sizes in each of these subsequences include unique signatures that can be associated with phonemes of the spoken language. This attempt is an extension of a previous work done by Wright et al. [90], where packets length are used to extract phrases of a VoIP conversation.

In [55] a low cost attack is proposed that is capable of linking potentialy unrelated streams to the their initiator, hence reducing the anonymity provided in Tor. In [78] the authors identify packet traces by injecting a watermark to the inter-packet timing of the traffic.

3.2.1 Website Fingerprinting

Some of the earliest work specifically focussed on attacks and defences for encrypted web traffic appears to be that of Hintz [36], 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 [67] 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* [5] 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 [5], making use of packet direction and size information [59] and assuming that the packet stream has already been partitioned into individual web page fetches. For example in [74] the timing information is not considered in the feature set, hence the attack can be countered with defences such as BuFLO in [16] leading to a success rate of only 10%. In [33], the feature set includes packet count, distribution of packets *etc.* all of which can only be obtained when the beginning and end of transmissions are known. In [35, 46] Bayes classifiers based on the direction and size of packets are considered while in [60] an SVM classifier is proposed. In [47] classification based on direction and size of packets is studied using Levenshtein distance as the similarity metric, in [51] using a Gaussian Bag-of-Words approach and in [74] using K-NN classification. In [8] 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, [16] considers Bayes and SVM classifiers and finds that a range of proposed defences are ineffective. In [30] remote inference of packet sizes from queueing delay is studied.

3.2.2 Timing Attacks

In [37] the authors introduce an attack based on inter-arrival times which can classify the traffic of different applications. A method is proposed to remove noise from the time signatures. In [29] a side channel attack is proposed that investigates the RTT of ICMP messages.

3.2.3 Robustness of Traffic Analysis Attacks

To the best of our knowledge all fingerprinting attacks involve use of training data collected at the same link on which the attack is to be performed. However [29] considers a different type of attack where a remote adversary living in a different network than the victim, attempts to infer contents of their traffic by analysing the round-trip times (RTTs) of ICMP echo messages sent to the victim. Regarding web traffic attacks where the training data is collected at a different time from when the attack is performed, [5] considers an attack where the training samples are taken hourly over a period of three months. For 100 websites the accuracy of the attack using 24-hour training data is 23%, although this improves when the number of guesses are increased or the dataset is narrowed down to a small subset of the web sites. In [8] the training samples are collected using different platforms, but it is unclear whether the platforms are within the same network or if samples of the same web page are fetched over different platforms.

3.3 Defense Against Traffic Analysis

As mentioned before, with regard to traffic analysis and associated defences, previous studies have established that the packet size, count, transmission rate, inter-arrival times *etc.* can reveal significant information about a traffic flow, see *e.g.* [16, 20, 22, 79] and references therein. A number of defences have been proposed to counter these traffic analysis attacks, mostly addressing attacks that make use of packet size, count and the length of transmission *e.g.* they modify the traffic by padding the packets to be the same length or by adding dummy traffic to the end of the packet sequence to mask

the packet count [31]. These defences are, however, ineffective against attacks that solely use timing information of network flows, see for example [20] and [22]. Among the earliest defences against traffic analysis attacks is HTTPOS [48] and the work of Wright et al [91]. These modify packet size, packet timing and payload size and provide fairly successful defence against a number of classifiers [5, 11, 35, 46, 67, 93]. Similarly, In [16], BuFLO is proposed as a defence against traffic analysis attacks. The model combines previous countermeasures such as use of fixed packet sizes and constant rate traffic to respond to the attacks based on packet size information. Subsequently, CS_BuFLO in [6], made modifications to BuFLO based on rate adaptation, but this comes at the expense of a high bandwidth overhead. The Tamaraw defence in [7] also works similarly to the BuFLO concept, with the length of packets and length of traces as design parameters.

In [5] the authors alter the timing patterns by delaying ACKs which would add a transmission delay overhead to user experience. A similar defence is proposed in [54] where traffic traversing over Tor bridges is morphed to resemble a Skype video call making it difficult for adversary to distinguish the source.

In [26] a defence is proposed against timing attacks which obfuscates the traffic patterns by using Chaum's mix strategy. The defence aims to remove the link between client and server by minimizing the entropy between original and morphed traffic. Similarly to the present work, the idea of sending super-traces that represent a group of web pages is considered in [58]. However the features considered are based on packet size and frequency whereas here we consider the timing features of a packet sequence. More recently Wang *et al* [77] suggested a half-duplex transmission method where the client cannot send multiple requests to servers in parallel.

3.3.1 Tradeoff Between Privacy, Fairness and QoE

There exists a large body of research focused on fair rate allocation in shared networks but without consideration of privacy, see for example [4, 18, 50] and references therein. With regard to privacy, fairness has previously been considered in the context of mix networks [10] where the aim is to achieve unlinkability/anonymity. Mishra et al. [53] propose a proportional fair scheduling model that preserves source anonymity and in [52] investigate the trade-off between anonymity and quality of service. The focus in both papers is on hiding the identity of the users in a shared network.

To the best of our knowledge none of this previous work on defences against traffic analysis attacks has considered fairness or the joint aspect of the trade-off between privacy, throughput and delay in a shared network.

Chapter 4

A Traffic Analysis Attack Using Only Timing Information

4.1 Introduction

In this chapter we consider an attacker of the type illustrated in Figure 7.1. 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 the attacker relies solely on packet timestamp information whereas 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

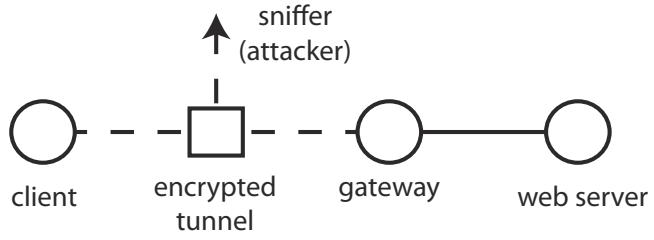


Fig. 4.1 Schematic illustrating attacker of the type considered. A client machine is connected to an external network via an encrypted tunnel (ssh, SSL, IPSec *etc.*). The attacker can detect the time when packets traverse the tunnel in the uplink direction, but has no other information about the clients activity.

currently either already available or being implemented in a number of popular VPNs [8]. Secondly, alternative attacks based on packet counting [8, 16] 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.*) [37], to our knowledge, there is no previous work reporting use of timing information alone to construct a successful attack against encrypted web traffic.

The main contributions in this chapter 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 58% 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 to the attacker.

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 [16] 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 to research.

4.2 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. Our experiments on use of uplink, downlink and uplink+downlink traffic suggest that downlink traffic provides no additional information regarding timing patterns over uplink traffic. The reason is that the timing of ACKs in uplink traffic is correlated to that of downlink packets which means that using only 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, while the corresponding downlink traffic could be morphed using inter-flow transformations *e.g.* flow mixing, split and merge [78]). We therefore focus on an 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.

Figure 4.2 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 4.2 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

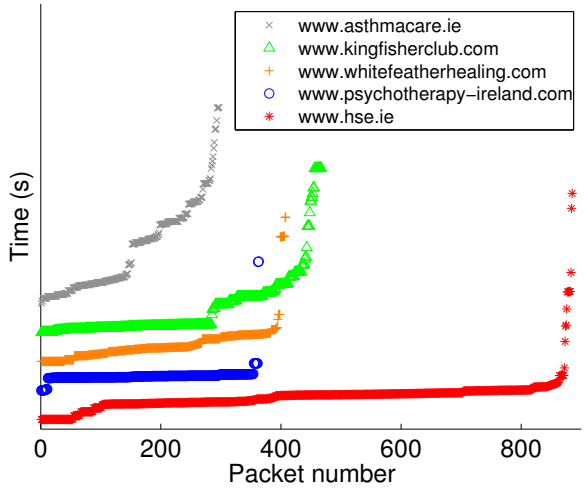


Fig. 4.2 Time traces of uplink traffic from 5 different Irish health-related web sites are shown. It can be seen that the web site time traces exhibit distinct patterns. The traces are shifted vertically to avoid overlap and facilitate comparison.

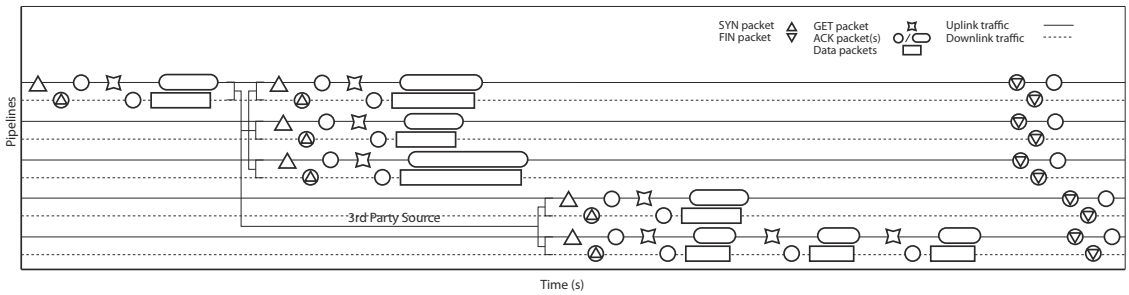


Fig. 4.3 This figure represents a typical web site query. It starts by requesting the index page. Then as the browser parses through this page more objects are fetched in parallel. Some objects may also be outsourced to 3rd party web sites which have their own pipelines. Dynamic content may be updated at intervals, as indicated in the last two lines of the figure, and connections tend to close in groups.

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 Figure 4.3. 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 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.

4.3 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.

4.3.1 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

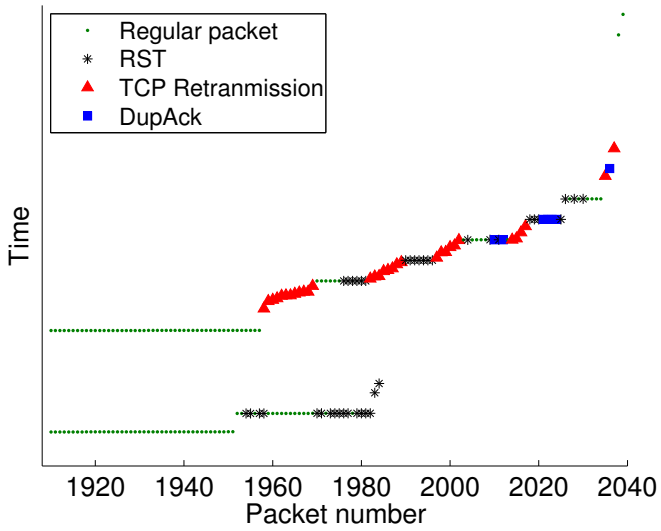


Fig. 4.4 Illustrating impact of changes in packet loss on the packet timestamp sequence. The bottom sequence shows the packet sequence at connection closing of a loss-free web fetch, while the top sequence shows the corresponding section from a different fetch of the same web page that was subject to packet loss and exhibits TCP retransmissions and DupACKs.

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.

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 4.4 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.

4.3.2 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) [44]. 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 $(p_{k+1}^i, p_{k+1}^j) \in V_{p_k^i, p_k^j} := \{(u, v) : u \in \{p_k^i, p_k^i + 1\} \cap \{1, \ldots, n\}, v \in \{p_k^j, p_k^j + 1\} \cap \{1, \ldots, m\}\}$, $k = 1, \cdots, l-1$. 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 4.5, 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 $\boldsymbol{p}^*(\boldsymbol{t}, \boldsymbol{t}')$ can be efficiently found using the backward recursion,

$$(p_k^i, p_k^j) \in \arg \min_{(p^i, p^j) \in V_k} C_{k+1} + c_{\boldsymbol{t},\boldsymbol{t}'}(p^i, p^j) \tag{4.1}$$

$$C_k = C_{k+1} + c_{\boldsymbol{t},\boldsymbol{t}'}(p_k^i, p_k^j) \tag{4.2}$$

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 (4.1), 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, in our data we found that this cost can lead to 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 [44] we instead use the following element-wise cost

$$c_{\boldsymbol{t},\boldsymbol{t}'}(p^i, p^j) = (D_{\boldsymbol{t}}(p^i) - D_{\boldsymbol{t}'}(p^j))^2 \tag{4.3}$$

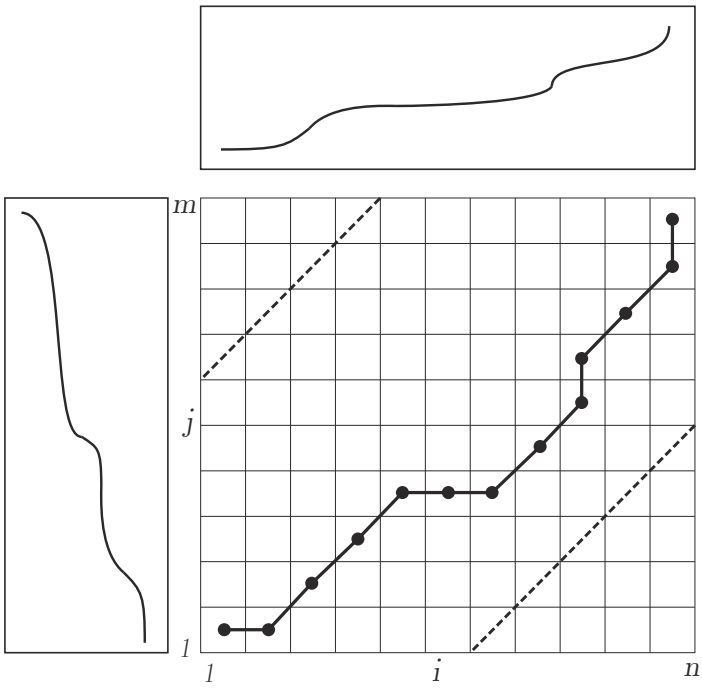


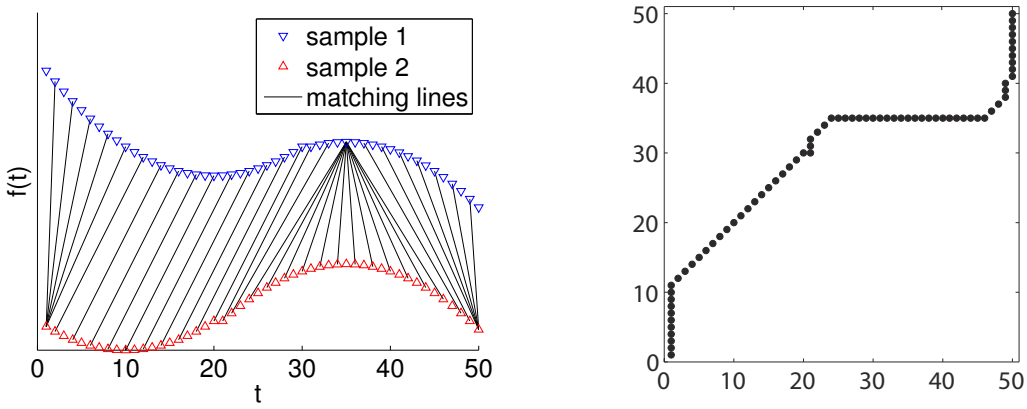
Fig. 4.5 Illustrating a warping path. The dashed lines indicate the warping window.

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 4.5) by setting $C(\boldsymbol{p}) = +\infty$ for paths $\boldsymbol{p} \in P^l_{mn}$ for which $|p^i_k - p^j_k| > \max\{w \min\{n, m\}, |m - n|\}$ for any $k \in \{1, \cdots, l\}$.

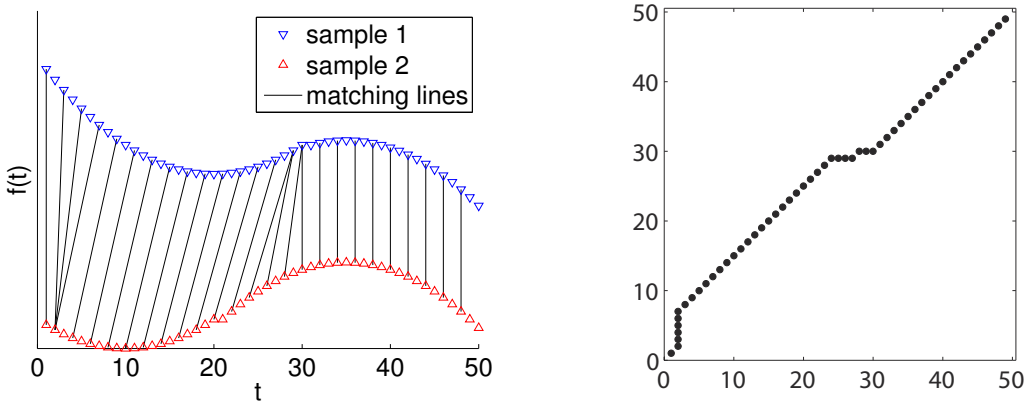
Figure 4.6b illustrates the alignment of points between two sequences obtained using this approach and for comparison Figure 4.6a shows the corresponding result when using Euclidean cost. The figure shows the warping paths on the right-hand side and an alternative visualisation of the mapping between 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 [44] 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.

4.3.3 *F*-Distance Measure

Given two timestamp sequences, the warping path is a mapping between them. With reference to Figure 4.5, sections of the warping path which lie parallel to the diagonal



(a) Euclidean cost



(b) Derivative cost

Fig. 4.6 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.

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 $\boldsymbol{p} = \{(p_k^i, p_k^j)\}$, $k = 1, \cdots, l$ 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_k^i or p_k^j remain constant and we count the subpaths which are longer than one. For example, for the setup shown in Figure 4.7 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

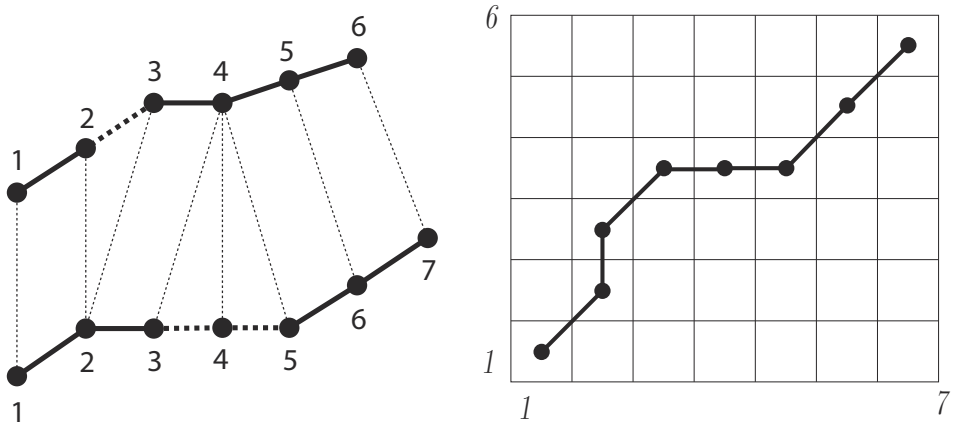


Fig. 4.7 Illustrating method for calculating the F-distance between two timestamp sequences.

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 4.7.

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:

$$\phi(\boldsymbol{t}, \boldsymbol{t}') := \frac{\sum_{\substack{s \in \{1,\ldots,r-1\} \\ \kappa_{s+1} - \kappa_s > 1}} \kappa_{s+1} - \kappa_s}{n + m} \tag{4.4}$$

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 4.5). For the example in Figure 4.7 the F-distance $\phi(\boldsymbol{p})$ is $(2 + 3)/13 = 0.385$.

4.4 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 entire project including codes, scripts and datasets for all measurement campaigns is available at [19]. The first 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 prune the pages that fail to load and then for each of the top 100 sites we carry out 100 fetches of the index page yielding a total 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 five days. In these datasets the browser cache is flushed between each fetch so that the browser always starts in a fresh state. In addition, a third dataset was collected consisting of the same 10,000 web fetches but now without flushing of the browser cache between fetches. The web pages were fetched over a period spanning November 2014 to January 2015. A `watir-webdriver` 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 only packet timestamps on the uplink were actually used.

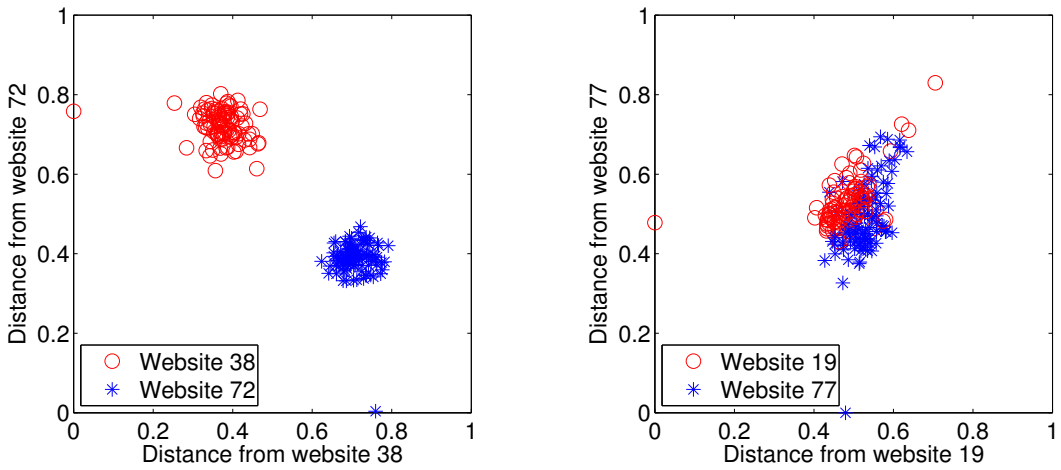
4.4.1 Hardware/Software Setup

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

4.4.2 Classifying Measured Timestamp Sequences

We use the F-distance measure $\phi(\cdot,\cdot)$ described in Section 4.3 to compare measured uplink timestamp sequences, with windowing parameter $w = 0.2$ unless otherwise stated.

Figure 4.8 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



(a) Distinct pages (b) Similar pages

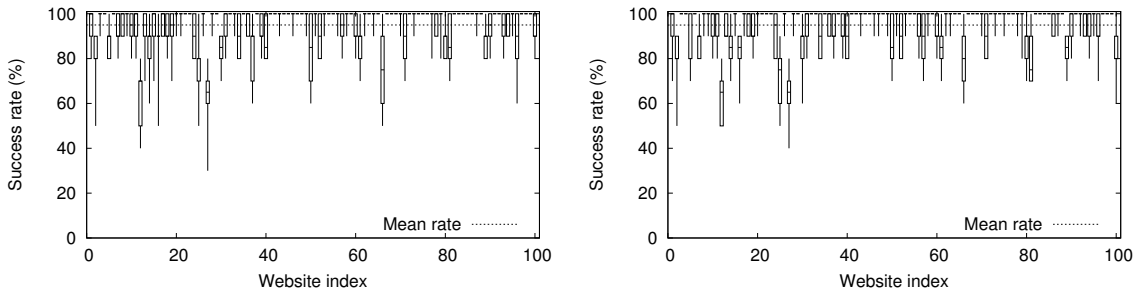
Fig. 4.8 Scatter plots for 4 different web pages using F-distance measure ϕ. In (a) two relatively distinct web pages are compared while the web pages in (b) are relatively similar.

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 4.8 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 4.8a 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 4.8b 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.

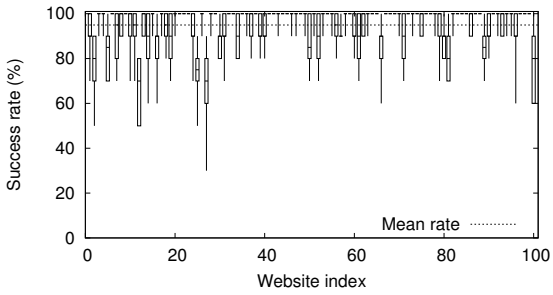
We considered two approaches for using $\phi(\cdot, \cdot)$ to classify timestamp sequences: K-Nearest Neighbours and Naive Bayes Classification.

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 3 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.



(a) $K = 1$ (b) $K = 3$



(c) $K = 5$

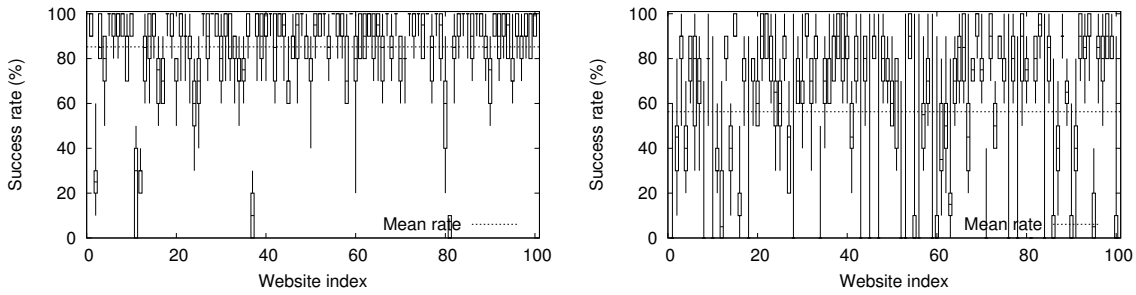
Fig. 4.9 K-Nearest Neighbours classification performance, no browser caching.

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.

4.4.3 Experimental Results

We begin by presenting results for the dataset where pages are fetched consecutively and the browser cache is flushed between fetches. Figure 4.9 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.



(a) Minimum mean (b) Minimum variance

Fig. 4.10 Naive Bayes classification performance, no browser caching.

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 95.01%, 94.97% and 94.98% respectively. These results for uplink traffic compare to a maximum success rate of 92.5% when using packet timestamps on the downlink for the classification, indicating that use of uplink or downlink timestamps has little effect on the performance of this classification attack. The results are also compared for a subset of 50 web sites selected randomly from the current 100, see Table 5.2, which also confirms that the effect of population size is minor.

For comparison, the success rates when web pages are fetched hourly over 5 days are 90.88%, 90.72% and 90.74%. Observe that there is a small (about 5%) reduction in success rate, which we assume is associated with the time-varying nature of some of the web sites. We discuss the effect of content and speed variability on the performance in Section 4.5.

Figure 4.10 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 85.2% and 56.3% 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 this chapter.

4.4.4 Standard vs. Cached: Different Versions of the 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

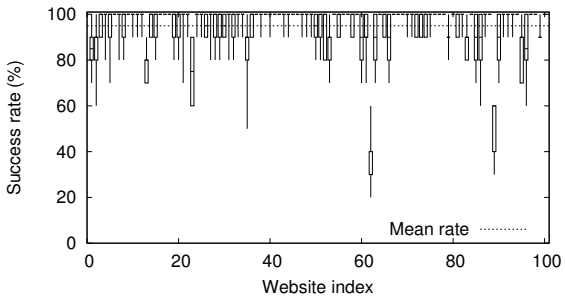


Fig. 4.11 K-Nearest Neighbour classification performance, with browser caching using 3 exemplars for each site. $K = 5$.

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 4.11 presents the measured classification accuracy when browser caching is enabled. This data is for the case where requests that reply with NOT MODIFIED use the cached content, which is probably the most common form of caching used in practice. It can be seen that regardless of the small size of the network traffic in this setup, the overall success rate for identifying web pages remains in excess of 95%.

4.4.5 Web Pages Outside the Training Set

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 need not hold, *i.e.* the user might have fetched a web page outside of the adversary's training

database, then we can use the following approach to first classify whether a measured packet timestamp sequence $\boldsymbol{t}$ is associated with a web site in the training set or not.

Recall that, as discussed in Section 4.4.2, for each web page i in the training set we have 3 exemplar packet timestamp sequences that are used for K-Nearest Neighbour classification. Given a packet timestamp sequence $\boldsymbol{t}$ we use K-Nearest Neighbour classification to estimate the nearest web page $w(\boldsymbol{t})$ within the training set and let $F_{min}(\boldsymbol{t})$ denote the minimum F-distance between the exemplars for this web page and the measured timestamp sequence. We can then use this value as the basis for a simple classifier. Namely, when $F_{min}(\boldsymbol{t})$ is greater than a specified threshold (which may depend on $w(\boldsymbol{t})$) then we estimate $\boldsymbol{t}$ as lying outside the training set, and when $F_{min}(\boldsymbol{t})$ is below the threshold then we estimate $\boldsymbol{t}$ as lying within the training set. It remains to select an appropriate threshold for each web page in the training set.

For every timestamp sequence $\boldsymbol{t}$ in the training set Figure 4.12 plots the distribution of $F_{min}(\boldsymbol{t})$ vs the index of the web site for which $\boldsymbol{t}$ is measured. This figure is a box and whiskers plot with the min, max and quartiles shown. For every web site we then remove its data from the training set and repeat the calculation. The distribution of these values is also shown in Figure 4.12. It can be seen that, unsurprisingly, the F-distance is consistently higher when a web site is excluded from the training set. We select the threshold for classification to try to separate these two sets of values. Namely, we take the average of the x percentile of the lower values and the $(100-x)$ percentile of the upper values as our threshold, where $0 \leq x \leq 100$ is a design parameter.

The classification error rate vs the threshold parameter x used is shown in Figure 4.13a. Two error rates are shown, firstly the fraction of web pages which are outwith the training set but which are classified as lying within it (which we refer to in this section as false positives) and secondly the fraction of web pages which are within the training set but which are classified as lying outwith it (which we refer to as false negatives). The standard deviations of these error rates across the web pages is also shown in Figure 4.13b. It can be seen that thresholding with $x = 90$ yields equal error false negative and false positive rates of about 8.0%, which is close to the complement of the reported success rate reported in the preceding section.

4.5 Effect of Link Speed and Content Change on Classification Performance

By looking closely at performance of websites, it can be seen that the total mean success rate obtained in each measurement campaign is not monotone amongst in-

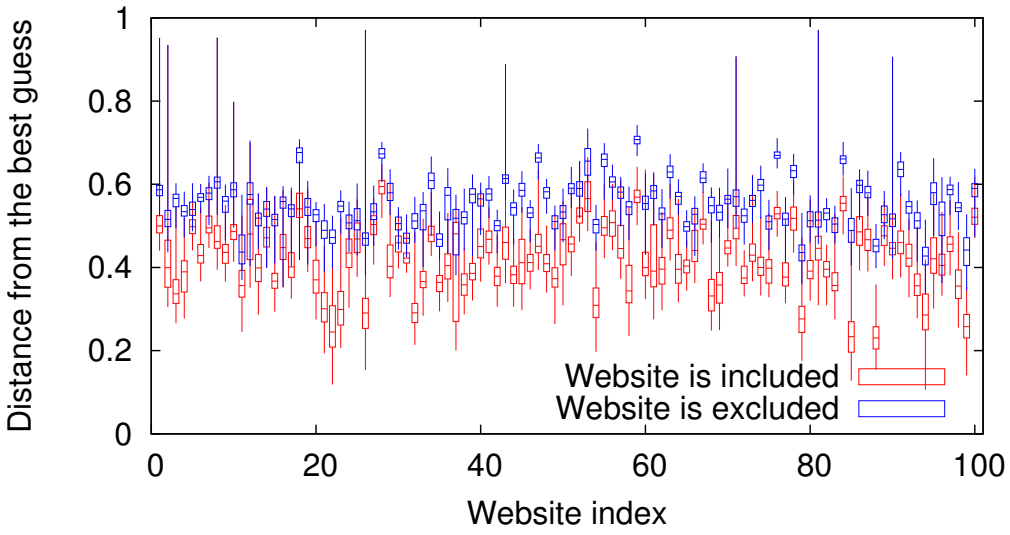
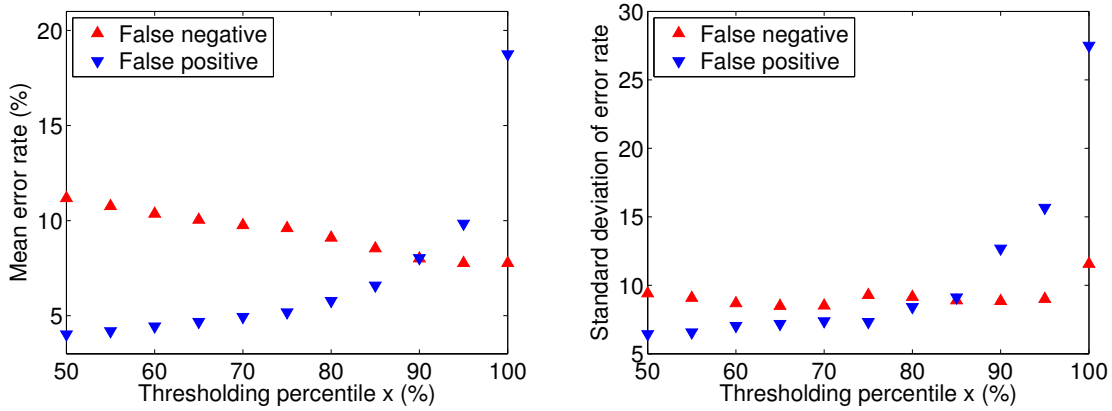


Fig. 4.12 Distribution of the F-distance between the measured packet timestamp sequences in the training dataset and the exemplar packet sequences for the best guess. Data is shown for when sequences of each web site are within the training dataset and for when they are removed. Ethernet channel, no browser caching.

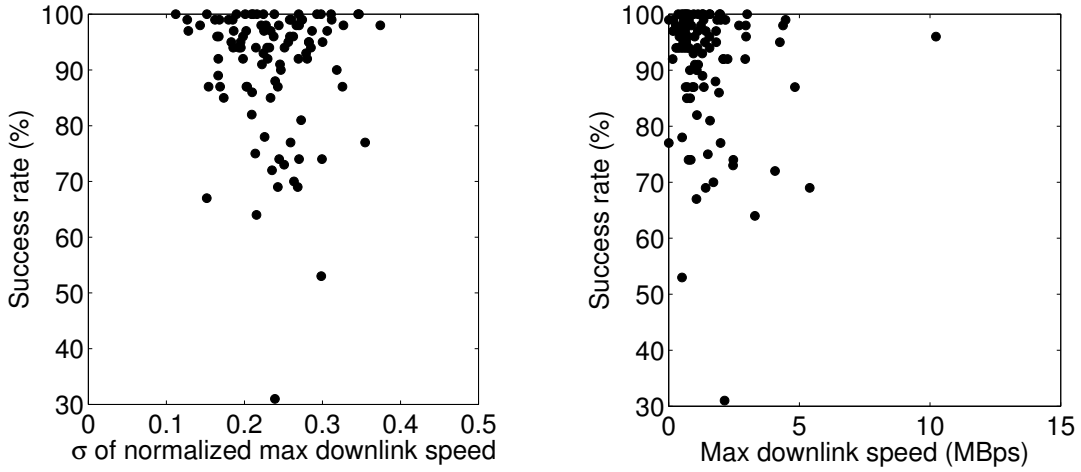


(a) Mean

(b) Standard Deviation

Fig. 4.13 Mean and standard deviation of false negative and false positive error rates vs the choice of F-distance threshold (specified via design parameter x).

dividual websites. In this section, we investigate possible reasons behind the poor performance of certain websites. We use the same ethernet dataset from Section 4.4.3 where samples are fetched hourly over 5 days.

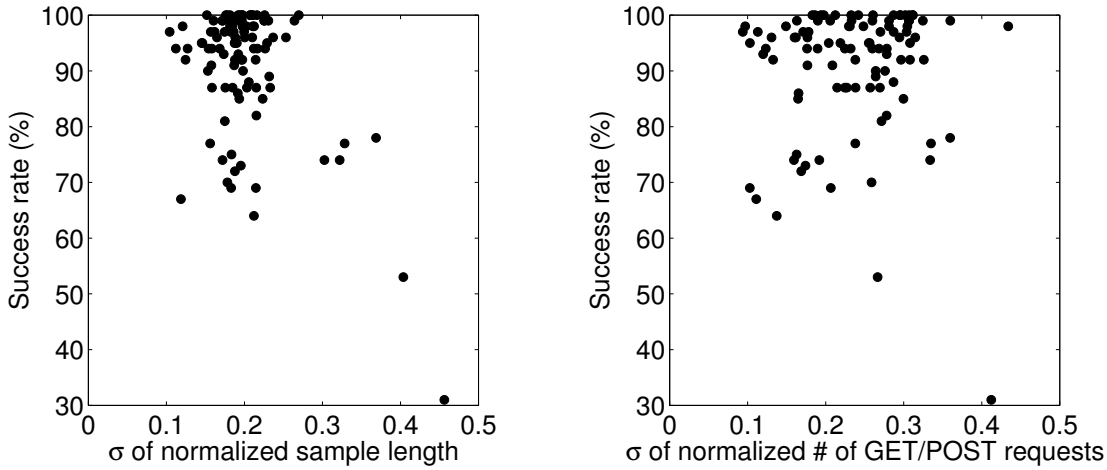


(a) Standard deviation (b) Median

Fig. 4.14 Scatter plot of max link speed standard deviation and median against success rate. Samples are taken hourly for 5 days over ethernet channel.

1. *Network Speed.* The link speed between the client and each web server varies from one website to another. It is also different between samples of the same page. To investigate the effect of network speed on the classification performance, we calculated peak downlink speed during each fetch (the results for uplink and uplink+downlink speed are similar). Then in order to compare the metrics, values v_i for samples of each page are normalized by $\max(v_i) - \min(v_i)$ and their variance is evaluated. Figure 4.14a illustrates the scatter plot of normalized standard deviation of link speed against success rate of each website. It can be seen there is no strong correlation between these two metrics that would suggest that a web site with more variable link speed should result a lower success rate. Figure 4.14b shows a comparison against median speed for each web site.

2. *Sample Length and GET/POST Request Count.* For each web page we plot the standard deviation of the normalized number of uplink packets (a measure of the variability of the web page over time) and the corresponding success rate (see Figure 4.15a). The results for uplink and uplink+downlink are similar. We also plot success rate against the maximum number of GET/POST requests for each website (Figure 4.15b). It can be seen that, there is no strong correlation



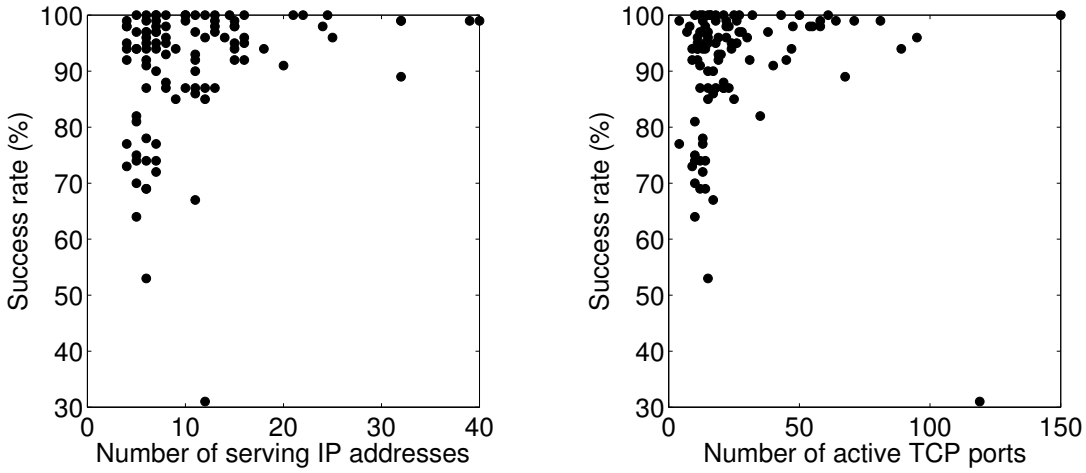
(a) Sample length (b) GET/POST count

Fig. 4.15 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.

between these metrics and success rates which is suggestive that the classification attack is fairly insensitive to variability of web page content over time.

3. *IP Connections, Active TCP Ports.* In order to investigate the robustness of the attack against parallel connections, for each web site we plot the median number of serving IP connections and active TCP ports against their corresponding success rates. As illustrated in Figures 4.16a and 4.16b, again there is no clear correlation between these metrics and success rate which is suggestive that the number of active IPs/ports for each web site, which represents the number of parallel connections, has little effect on the performance of our attack.

The above results suggest that there is no strong correlation between the performance of our attack and the link speed, minor content change or the number of parallel connections. However the choice of exemplars is essential to the performance of the attack. In particular when the content change is more than some threshold, the difference between samples can no longer be ignored by the attack. An example of this behaviour can be seen for website #10 in Figure 4.17. Two different versions of this page were observed during the experiment. As a result, one exemplar represents one version while the two other exemplars represent the other version of the page. This causes the K-NN method to fail to collect enough votes for a successful classification, which in turn leads to a success rate of only 31%.



(a) Open IP connections (b) Active TCP ports

Fig. 4.16 Scatter plot of median open IP connections and active ports count against success rate. Samples are taken hourly for 5 days over ethernet channel.

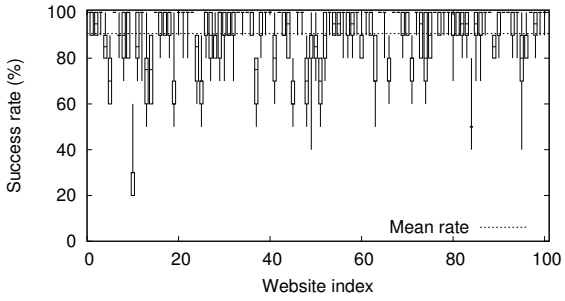


Fig. 4.17 K-Nearest Neighbours classification performance for samples taken once every hour for 5 days, no browser caching, $K = 5$.

Similar behaviour is observed when larger objects in a web page are fetched from different CDNs. The difference in delay between a client and CDN servers is mitigated by the current attack, since it only accounts for a single-point anomaly in time signatures. However if the link speed is different, traffic patterns vary for versions of web pages retrieved from different CDNs. This affects the performance of our classification for larger objects which are responsible for longer patterns in traffic signatures.

To overcome this issue, separate sets of exemplars are required to represent each version of a web page.

4.6 Finding a Web Page within a Sequence of Web Requests

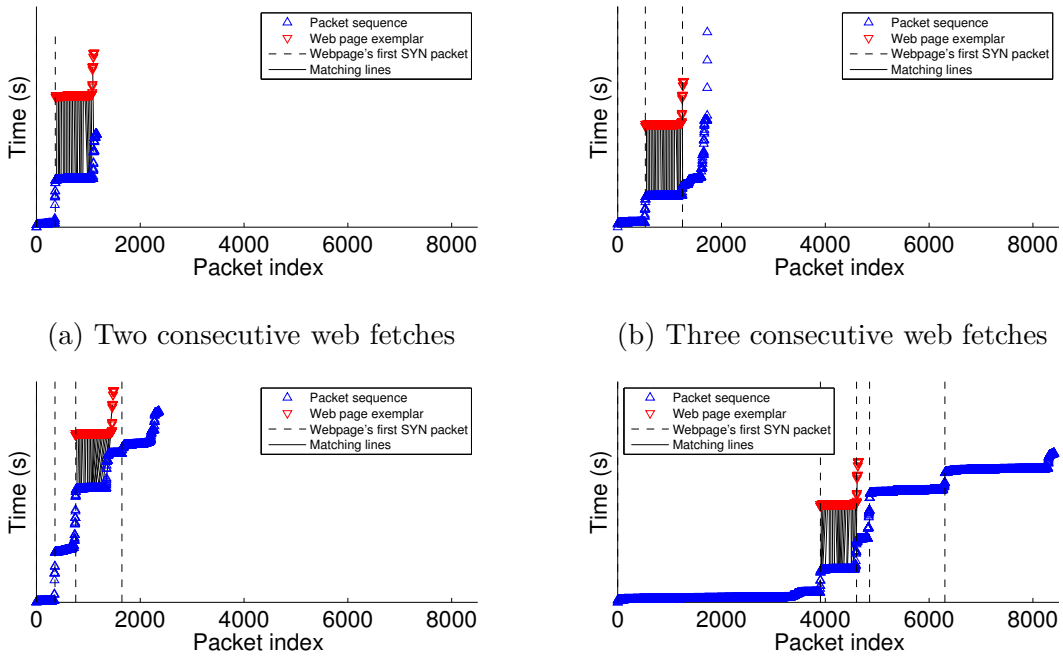
In the experiments presented so far we have assumed that within the observed packet timestamp stream the boundaries between different web fetches are known. This is probably a reasonable assumption on lightly loaded links where the link is frequently idle between web fetches. However, not only might 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 (plus 10 to cover the step size) 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 4.4.5 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).

4.6.1 Results

We constructed a test dataset as follows. For each run we pick one of the 100 web sites to be the target. We then uniformly at random pick up to 4 other web sites from the remaining web sites. The selected web sites are then permuted randomly and fetched one after another with a pause after each fetch acting as a “thinking period”. The



(a) Two consecutive web fetches

(b) Three consecutive web fetches

(c) Four consecutive web fetches

(d) Five consecutive web fetches

Fig. 4.18 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.

maximum time allowed for each fetch to complete is 25 seconds *i.e* the length of each pause is selected uniformly at random from 5-25 seconds. Repeating this for all web sites in the dataset, we created 100 test runs.

No. of consecutive pages	2	3	4	5
Ethernet	82%	80%	66%	64%
Femtocell	85%	82%	72%	70%

Table 4.1 Success rates of locating web pages among 2-5 fetches.

Using the classification approach described above we attempted to identify the location within each packet stream. Figure 4.18 presents four examples 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 web sites are summarized in Table 4.1 for both ethernet and femtocell links. With this approach, on an ethernet link we achieved a maximum success rate of 82% for locating the target web page within each packet stream within a position error of $w.l_s$ packets, where w is the window size at

which DTW operates (0.2 in our setting) and l_s is the average length of the 3 exemplars which are determined for each web site s separately. Given the limited information being used, these are remarkably high success rates and indicate the power of the timing-only attack. However, it can be seen that the success rate starts to lower as the number of consecutive fetches grows which leads to a longer packet stream that can potentially include similar patterns to the target web page. Moreover web pages with shorter length are less likely to be located properly due to their shorter signatures which are more likely to appear in the middle of a larger web trace.

4.7 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 ethernel channel, achieving mean success rates in excess of 90%.

Study of downlink and a preliminary study of uplink+downlink traffic suggest little difference from the uplink results presented in this chapter, since the timing patterns of the uplink and downlink packets are strongly correlated. Moreover, the proposed attack proves to be robust against different link speeds, different numbers of parallel connections and minor content change, being able to maintain an overall success rate of 91% for measurements collected over the course of 5 days.

Since this attack only makes use of packet timing information it is impervious to existing packet padding defences. 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.

Chapter 5

Robustness of the Traffic Analysis Attack

5.1 Introduction

In previous chapter we proposed a traffic analysis attack that makes use of only timing information of packets. We evaluated the attack over an ethernet channel. The results of detecting a web page in a sequence of packets were also presented. The experiments in Chapter 4 were carried on over a stable and rather clean ethernet channel, where the training and test data were collected on the same link and around the same time. In this section we evaluate the performance of the proposed attack against more challenging network conditions. We also study the robustness of the attack when training data is collected on a different link up to 50km away and at a time up to 16 weeks apart from the user queries.

5.2 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.

5.2.1 Femtocell Traffic

We present measurements of web page queries carried out over an encrypted femtocell channel and evaluate the accuracy with which the web page being fetched can be inferred using only packet timing data. 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.

Similar to previous experiments, the dataset consists of 100 fetches of the home pages of each of the top 100 Irish health, finance and legal websites as ranked by www.alexa.com under its Regional/Europe/Ireland category in September 2014, yielding a total of 10000 individual web page fetches. The web pages were fetched during February 2015 where all samples of each website are fetched consecutively over an hour. To study the effect of time on dynamic pages, a second data set is collected on May 2015 where this time, for each website a sample is taken every hour for a duration of 6 days. A `watir-webdriver` 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 femtocell connecting the client to the network although only packet timestamps on the uplink were actually used. Further experiments were also carried out for the downlink, and for different versions of websites (caching *etc.*) to explore their impact on the results.

Hardware/Software Setup

The client computer is the same Sony laptop used for the ethernet measurements. It now uses a Huawei K3770 HSPA USB Broadband Dongle to connect wirelessly to the internet via a Femotcell. The femtocell is 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 which is running on a AMD Athlone 64 X2 Dual Core Proc 5000+ CPU and 4GB memory is also connected

to this hub and logs all packets. The client and monitor computers both run Ubuntu Linux 14.04 LTS Precise.

Results

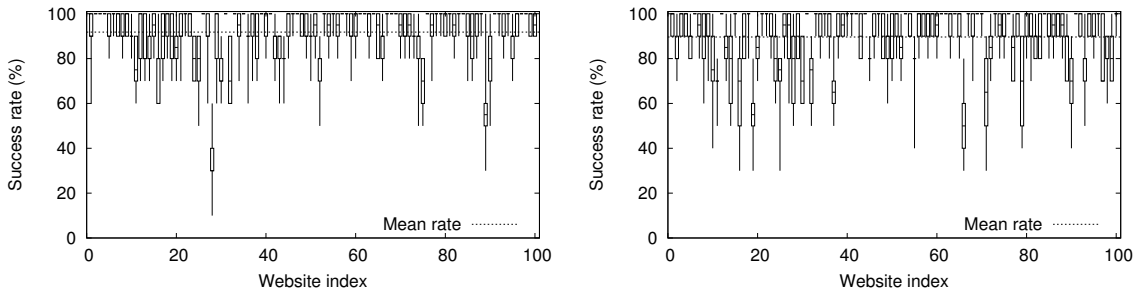
In contrast to the relatively clean ethernet channel considered in Section 4.4.3, we found that traffic passing over the wireless femtocell link is often distorted by factors such as wireless and cellular noise, encoding/decoding delays, cellular control plane traffic *etc.* These distortions typically appear as shifts along the x-axis of the packet timestamp patterns and/or as delays in the y-axis. For each of the 100 web pages studied, Figure 5.1 details the measured classification accuracy using the K-NN approach (use of a naive Bayes classifier was also investigated, but the K-NN classifier was found to consistently offer better performance). The mean success rate is 91.8%. This compares with a baseline success rate of 1% for a uniform random classifier over 100 web sites and so is likely to represent a significant compromise in privacy. For comparison, when 50 rather than 100 web sites are used the success rate is 94.9%, indicating that our results are relatively insensitive to the number of web pages. A mean success rate of 89.7% is obtained when the samples are fetched every hour over a duration of 6 days. The insignificant change in classification accuracy shows the effectiveness of the attack even on dynamic websites whose content evolves over time.

The mean success rate of 91.8% is also compared with 90.5% for when the attack is conducted against downlink traffic, which shows little difference in success rates. Moreover to show the effect of caching, a smaller but similar experiment is conducted, (using the remainder of our data allowance) where the caching is enabled on the browser and the success rate is 86.7%. The results are summarized in Table 5.1.

The classification results for the femtocell is comparable with that of an ethernet channel where the traffic is routed from the client to the internet, where as already noted a mean success rate of 95.0% was obtained. It can be seen that using a mobile broadband with a femtocell device makes little difference to the classification accuracy compared to normal network use. The measured performance for other parameter settings is also summarised in Table 5.1.

Effect of Load Variability on the Performance of the Attack

The femtocell results show the robustness of our proposed attack over wireless links. However if the network speed is affected by load variability (due to handovers on cellular links, a change in number of users of a Wi-Fi access point, etc.), the performance of the attack may be affected as the traffic patterns in training and test data may



(a) Consecutive fetching over one hour (b) One fetch every hour for 6 days

Fig. 5.1 K-Nearest Neighbours classification performance when using mobile broadband over a femtocell, no browser caching, $K = 5$.

Channel	Traffic Type	Population Size	K		
			1	3	5
Femtocell (Consecutive Fetch)	Uplink	100	92.6%	91.8%	91.83%
	Uplink	50	95.72%	95.20%	94.86%
	Downlink	100	91.80%	90.95%	90.50%
	Cached	25	91.48%	89.92%	86.68%
Femtocell (Hourly Fetch)	Uplink	100	90.2%	90.08%	89.65%
Ethernet (Consecutive Fetch)	Uplink	100	95.01%	94.97%	94.98%
		50	97.16%	97.18%	97.04%

Table 5.1 Summary of the measured success rate of the timing-only attack for femtocell using 3 exemplars.

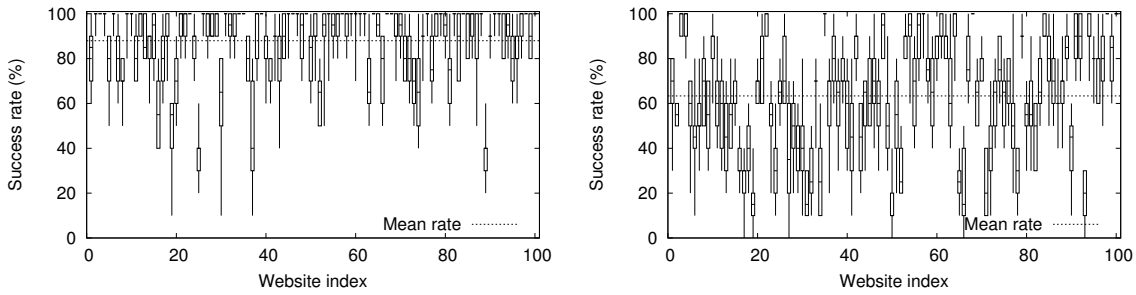
vary. Such pattern alterations are generally ignored by our proposed DTW distance, unless the difference is significant, e.g. when training data is collected over a congested channel and test data over a normal one or vice versa.

5.2.2 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.

Hardware/Software Setup

The experimental setup is identical to that used in Section 4.4 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



(a) Slot size: 1ms (b) Slot size: 10ms

Fig. 5.2 Time-slotted tunnel K-Nearest Neighbours classification performance, no browser caching, $K = 5$.

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 which captures them using the PREROUTING hook, extracts the payload and forwards this to the application layer.

Results

Figure 5.2 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 88% when the tunnel slot size is 1ms and 63% 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 95% 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 not proportional to the sacrifice we make in terms of delay and throughput (with such a large slot size we are capping the downlink throughput to 150KB/s). This approach therefore appears to be unappealing as a practical defence against the timing-based attack considered here. Of course more sophisticated types of defence may be more effective, and we will consider defences further in a later chapter.

5.2.3 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 [14, 69]. A large body of traffic analysis attacks and defences have been proposed for Tor [28, 38, 41, 54, 55, 75]. Our attack directly targets one of the primary objectives of the Tor network, namely to ensure unlinkability between users and the servers which they visit. In contrast to correlation-based attacks, which seek to correlate packet timings across hops of the Tor network and require a strong adversary with a broad view of the network spanning many hops, our attack requires only access to the first network hop and so can potentially be employed by a considerably weaker adversary.

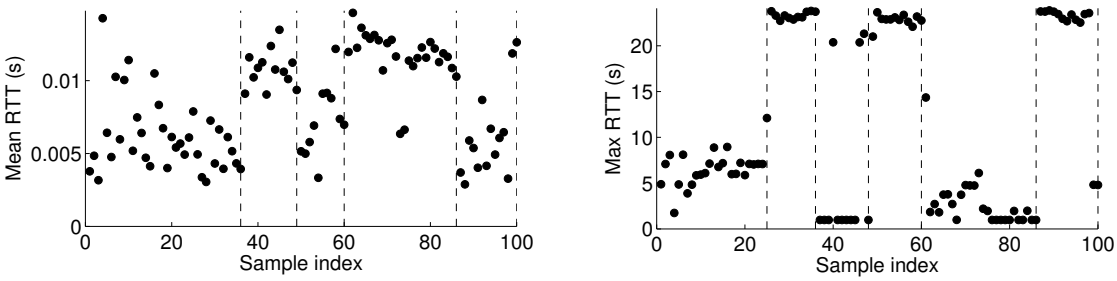
Hardware/Software Setup

The experimental setup is similar to that in Section 4.4 with web site samples captured in September 2014 except that the traffic from the client browser, Mozilla Firefox 36.0 is proxified over Tor v0.2.5.11. Note that we also explored use of the Tor browser but found that a significant subset of the web sites failed to load, timed out or required a CAPTCHA to be solved for each page fetch which created complications when scripting fetches. We also investigated using Firefox with Tor pluggable transports (such as obfs4 *etc.*) but we found that using these add-ons had a huge impact on delay such that most web sites fail to load even after 5 minutes. As before, the browser cache is flushed between fetches.

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 5.3, 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



(a) Mean RTT for packets of each sample (b) Max RTT for packets of each sample

Fig. 5.3 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).

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 5.3b. These changes persist for a period of time as Tor only performs rerouting periodically.

Figure 5.4 illustrates the impact of Tor on the packet timestamps measured during a web page fetch.

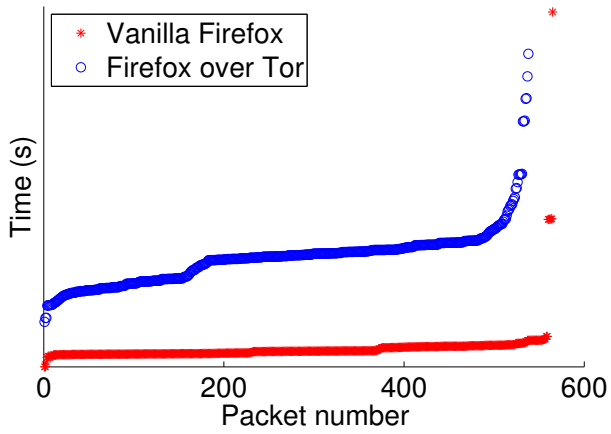


Fig. 5.4 Time traces of uplink traffic measured when fetching www.medicalcouncil.ie. Measurements are shown both when using vanilla Firefox and when using Firefox with the Tor plugin.

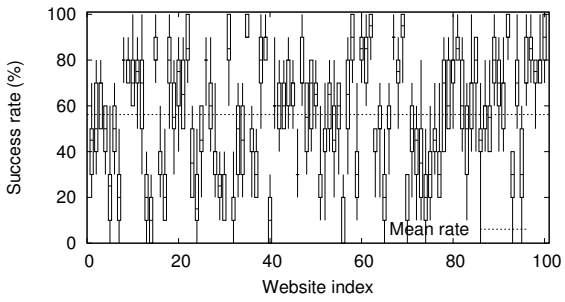


Fig. 5.5 Tor network K-Nearest Neighbours classification performance, no browser caching, $K = 5$.

Results

Figure 5.5 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 56.2% which compares with the mean success rate of 95.0% when using a clean ethernet channel. As might be expected, use of the Tor network significantly reduces classification accuracy. However, the success rate of 56.2% compares with a baseline success rate of 1% for a random classifier over 100 web sites and so still is likely to represent a significant compromise in privacy. We note also that this compares favourably with the 54.6% rate reported by Panchenko *et al* in [60] against Tor traffic using packet size and direction information.

5.2.4 Other Proposed Channels

A number of other channels have been proposed in the literature as a defence against traffic analysis attacks. Wright *et al* [91] suggest a traffic morphing method which maps the packet sizes of one web site to the packet distribution of another site. This defence fails to overcome the attack considered here since it makes use only of timing information and does not use packet size information. This is also the case for all of the packet-size based defences proposed in the HTTPOS scheme introduced in [48]. A potential defence against timing attacks is to modify the packet timing pattern by delaying transmissions. However, although this might be expected to counter timing-based attacks such as that considered here, such defences will also have an impact on delay. For example, BuFLO introduced in [16] is similar to the time slotting method that we consider above but incurs a substantial impact on delay and bandwidth, with 190% bandwidth overhead reported in [74].

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%	-

Table 5.2 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.

5.3 Robustness of the Attack Against Distance and Time Lapse

For the remainder of this chapter, we consider an attacker of the type illustrated in Figure 5.6. A client A browses the web over an encrypted connection, *e.g.* a VPN, and the attacker can observe the encrypted traffic. As before, it is assumed that all encrypted packets are padded to be the same size so that the packet size reveals no information about the nature of the traffic and the only information available is the timing pattern of the traffic traversing the link. The attacker's objective is to guess, with high probability of success, the web sites being visited by the victim.

To assist with the attack the attacker can, of course, themselves fetch web pages of interest and record the packet timings for use as training data (against which the victims data can be compared). An attack of this type making use of training data collected over the victims link was considered in the previous chapter and it was demonstrated that the web pages being browsed could indeed be accurately inferred with high probability, achieving mean success rates in excess of 90% for both wired and wireless traffic. However, it is often relatively difficult to collect such training data in an unobstrusive manner since it takes time and creates a significant volume of traffic over the victims link. In this chapter we therefore consider the feasibility

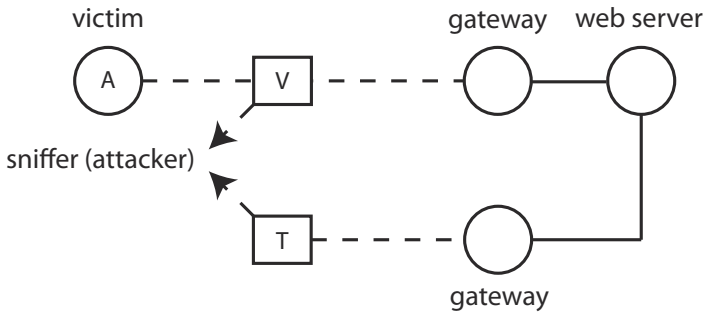


Fig. 5.6 Schematic illustrating attacker of the type considered. The targeted victim is connected to the Internet via an encrypted tunnel (ssh, SSL, IPSec *etc.*) shown by dashed lines. The attacker can measure the timing of the victims packets traversing the tunnel at point V in the uplink direction. In addition, the attacker can make their own measurements over a second link T, different from that used by the victim.

of an attack based on training data collected over a link which is different from the victims. This might be located at a neighbouring house or office, but of more interest is whether training data can usefully be collected at a location much further away *e.g.* at a location within the same city as the victim but otherwise not in their vicinity. Such an attack would clearly be relatively easy to carry out and of significant concern.

We demonstrate that an attack which infers the correct web page >87% of the time is feasible even when the training data is collected at a distance of up to 25 km from the victim, provided that the type of link is similar *e.g.* if the victim link uses a cable modem then the training data should be measured over a cable modem link.

We further investigate the impact of distance in time between when the training data is collected and when the attack is performed. As might be expected, as the time elapsed increases the accuracy of the attack falls. However, the rate of decrease is surprisingly small: up to two months between collection of the training data and performance of the attack around 80% of web pages are estimated with >70% accuracy.

5.4 De-anonymizing Web Fetches Using Training Data from Different Locations

5.4.1 Measurements

We collected packet trace timing measurements at a variety of locations in Dublin and Maynooth (a town about 20 km west of Dublin) in Ireland. The hardware/software setup for the experiments in this section are similar to that explained in Section 4.4.1.

Web Pages

At each measurement time and location we fetched the 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. We prune the pages that fail to load and then for each of the top 100 sites we carry out 100 fetches of the index page yielding a total of 10,000 individual web page fetches in the data set collected at each time and location. Collection of these 10,000 pages was carried out over a period 4-5 days (so we collected more than 24 days of data across 6 locations). In these datasets the browser cache is flushed between each fetch so that the browser always starts in a fresh state. Fetches for different web sites are interleaved so that the measurements collected for a given web site are evenly spread over the duration of the measurement campaign at a particular location, to try to avoid bias due to changes in network conditions with time of day.

Place

We collected measurement datasets at 6 locations: (i) three university campus' namely, Trinity College Dublin, Dublin Institute of Technology and Maynooth University during 7-11 Jul, 24-28 Jun and 30 Jun-4 Jul 2015 respectively; (ii) three households, namely two in Dublin during 20-24 Jul and 5-9 Aug 2015, and one in Maynooth during 13-17 Jul 2015. For the campus measurements the target machine was connected via a gigabit ethernet LAN to the university network and then in turn to the Internet via a 10Gbps connection. For the household measurements the target machine is connected via ethernet to a cable modem and then to the Internet via a commercial Irish broadband network.

Training \ Test	DIT	MU	TCD	Home 1 (Maynooth)	Home 2 (Dublin)	Home 3 (Dublin)
DIT	95.31%	78.76%	68.08%	56.91%	60.13%	56.87%
MU	78.65%	90.98%	83.53%	51.04%	54.57%	48.87%
TCD	71.71%	85.34%	90.53%	48.75%	51.34%	46.09%
Home 1 (Maynooth)	64.69%	61.81%	57.9%	92.68%	89.04%	88%
Home 2 (Dublin)	65.89%	64.49%	58.99%	89.14%	94.20%	89.89%
Home 3 (Dublin)	63.93%	56.64%	53.03%	87.11%	88.86%	94.98%

Table 5.3 Summary of the measured success rate of the timing-only attack vs the location used to collect training data and the location used to collect test data (36 pairs of tests at 6 locations).

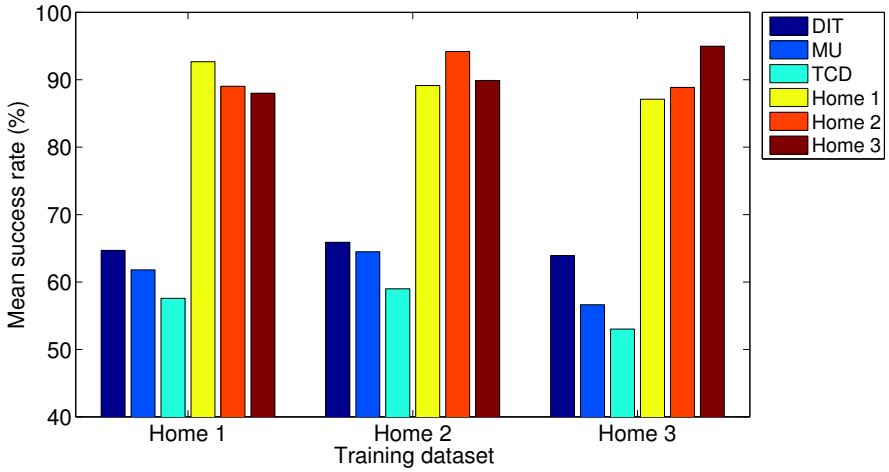


Fig. 5.7 Performance of the attack against different locations (specified in legend) when using training data from a home connection (specified on x-axis).

5.4.2 Results

Using the data measured at each location in turn as the training data, we evaluated the performance of the timing-only attack/classification method described in Chapter 4 when applied against the data measured at the remaining locations. The results obtained are summarised in Table 5.3. This data is also visualised in Figures 5.7 and 5.8.

It can be seen that when training data is collected from the same location at which the attack is performed in (the diagonal entries in Table 5.3) then the percentage of web pages which are correctly identified exceeds 90% at all locations. More interesting is that when measurements taken at a different home location are used as training data

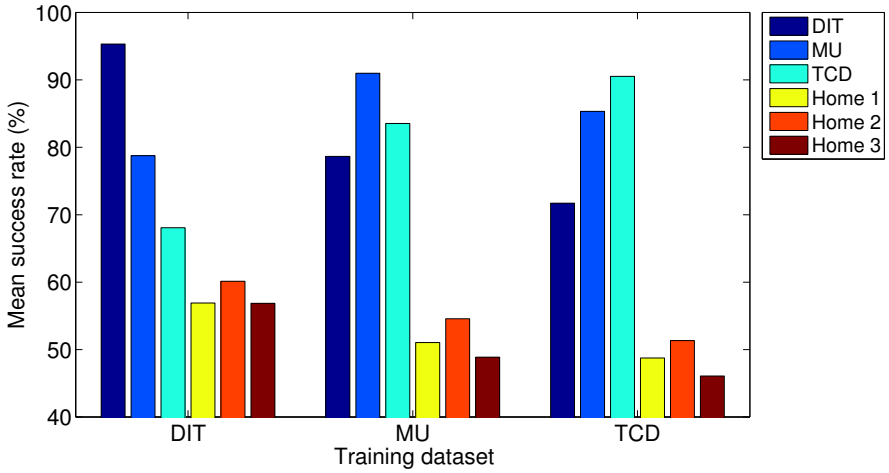


Fig. 5.8 Performance of the attack when using training data from a university campus connection.

for an attack at another home location then the impact on the success rate is less than 5% *i.e.* more than 87% of web pages are successfully identified regardless of where the training data is collected, so long as it is collected at a home location. Note that Homes 2 and 3 are spaced well apart (some 15km) in Dublin city while Home 1 is in a separate town some 20 km outside Dublin. While it might seem plausible that the differences in timing and network conditions would significantly degrade the performance of the attack, our data indicates otherwise and establishes that a successful timing-based attack based on remotely collected training data is indeed possible.

Observe, however, that it is important that training data is collected over a similar network connection to that being attacked. It can be seen from Figure 5.7 that when data collected on a home connection is used for training then the success rate for the attack against university campus connection falls to between 53% and 65%. Note that this is still quite a high success rate – the success rate expected when selecting a web page uniformly at random from the set of 100 pages studies is only 1% – but is significantly lower than the >87% observed when using a home connection for training.

Similar behaviour is observed for attacks against a home connection. It can be seen from Table 5.3 that when another campus connection is used for training then the success rates for university connections remain above about 70% whereas for the attack against a home connection the success rate falls to between 46% and 60%. This is also evident from Figure 5.8.

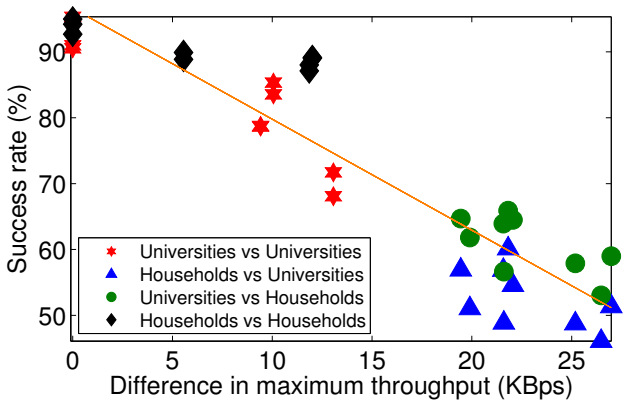


Fig. 5.9 Classification performance vs the difference (root mean square distance) in maximum throughput measured at training and test locations. Data is shown for the maximum uplink throughput but a similar correlation is also observed for the downlink and aggregate uplink/downlink throughput.

Observe that the success rate when using a remote university connection for training in an attack against a university connection is lower ($68-85\%$) than when using a remote home connection for training in an attack against a home connection ($87-89\%$). We inspected the data in more detail to try to better understand this phenomenon. Based on this analysis we found that a fairly strong correlation exists between the classification accuracy and the difference in the maximum throughput measured at the training and test locations, see Figure 5.9. Although this requires further investigation, it seems likely that consistent differences in network traffic load at the various university campus' is the source of the behaviour observed – we note that the campus' are significantly different in size and number of users[1]. For home connections, the scale of differences in traffic load might be expected to be lower both because the raw network bandwidth is less and because there are fewer users.

5.5 Impact of Elapsed Time on Performance

The foregoing results establish the feasibility of collecting training data at a different location from that where the attack is performed. In this section we investigate the

[1]Recall that we collected data for each web page evenly over each measurement campaign to control for differences in traffic load over the course of a day, and we find no evidence correlation between success rate and time of day.

Site \ Gap (weeks)	1	2	3	4	5	6	7	8
MU	92.56%	89.98%	88.62%	85.15%	85.03%	82.41%	79.15%	79.88%
TCD	92.17%	87.29%	77.29%	79.83%	78.81%	75.14%	72.88%	73.33%

Table 5.4 Measured success rates of the timing-only attack for Maynooth University and Trinity College Dublin ethernet from measurements taken over different time periods. (Every week from 29 Feb - 22 April 2016)

manner in which the elapsed time between collection of training data and performance of the attack affects its success. We expect the success of an attack to degrade as the time difference increases since many web pages are dynamic in nature and even small changes in news feeds or replacing images with new ones may well eventually accumulate and change the web page so much that the training data becomes less useful.

5.5.1 Measurements

University Campus

A new set of ethernet measurements is collected at Maynooth University and Trinity College Dublin. We use an identical hardware/software setup for both locations consists of two Dell Inspiron 5558 laptops with Intel Core i3-4005U CPU and 4GB of RAM and running Ubuntu 14.04 LTS Precise. The samples of each website are fetched once every hour over a period of 5 days from Monday to Friday for 8 weeks on the period of 29 Feb - 22 April 2016.

Femtocell

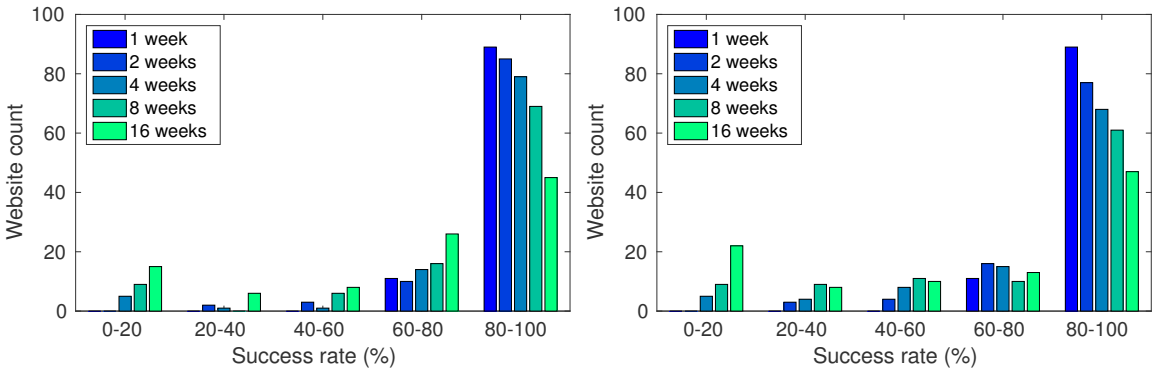
We also collected data over a third type of network connection, namely a wireless femtocell. The hardware/software setup for this experiment is similar to that explained in Section 5.2.1. Datasets were collected during 11-16 Feb and 6-11 May 2015.

5.5.2 Results

Using the data measured in Maynooth University and Trinity College Dublin at each time period in turn as the training data, we evaluated the performance of the timing-only attack/classification when applied against the data measured at the remaining time periods. The results obtained are summarised in Table 5.4. It can be seen that when the training data is collected around the same time as the attack is performed,

Training \ Test	FM 1	FM 2
FM 1	91.83%	43.79%
FM 2	47.03%	89.65%

Table 5.5 Measured success rates of the timing-only attack for femtocell over Maynooth University ethernet from measurements taken over different time periods. (Femtocell: 11-16 Feb (FM 1) and 6-11 May 2015 (FM 2).)



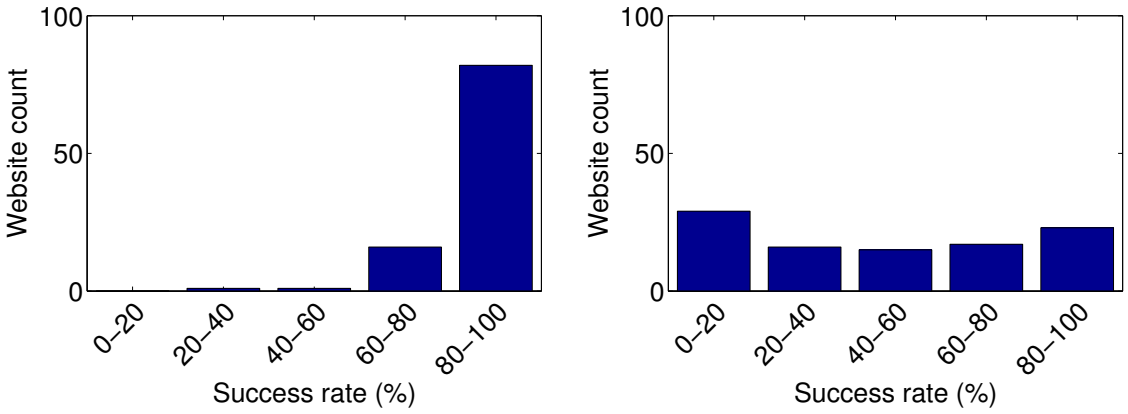
(a) MU (b) TCD

Fig. 5.10 Illustrating distribution of success rate over websites versus elapsed time between training and attack.

the success rate of the attack is >92%, in line with the results presented in the previous section. However, the success rate falls as the elapsed time between collection of the training data and performance of the attack is increased, as might be expected. The mean success rate lowers to 90%, 85.2% and 79.9% for Maynooth and 87.3%, 79.8% and 73.3% for TCD after 2, 4 and 8 weeks respectively. Note that while 73.3% is relatively lower than the 90% success rate obtained when using contemporaneous training data, it is still a relatively high value that would likely be of concern. That is, our data indicates that a viable attack may be carried out even with a fairly large elapsed time of 2 months between the collection of training data and performance of the attack.

Fig 5.10 shows this data in more detail, plotting the distribution of the success rate over the set of 100 web pages studied vs the elapsed time between training and attack. It can be seen even when the average success rate over all web pages falls vs elapsed time (see Table 5.4), a significant fraction of the individual web pages continue to be identified with high accuracy. We investigated this behaviour in more detail manually and find that for the web pages which experience a high drop in success rate either (i) the web site is down (returns 404 not found) in one test but not another, or (ii)

the web page has changed significantly (the number of GET requests and objects is significantly different).



(a) One week (b) Three months

Fig. 5.11 Distribution of success rates of individual web pages for different time gaps between training and test datasets. Datasets are collected from a femtocell channel running over Maynooth University campus internet.

For comparison, the results obtained using the measurements taken over a femtocell link are shown in Table 5.5. It can be seen that after 13 weeks have elapsed the success rate of the attack falls to less than 47%. The degradation in femtocell is significantly more than its ethernet counterpart. This is mostly due to its noisy and less cleaner channel which makes the classification more difficult over time. Figure 5.11 shows the corresponding distribution of success rates over the set of 100 web pages.

5.6 Summary and Conclusions

In this chapter we compare the performance of the attack against wired and wireless traffic, consistently achieving mean success rates in excess of 90%. Table 5.2 summarises our measurements of the success rate of the attack over a range of network conditions. Moreover, we show that time slotting is 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 of limited effectiveness.

We also consider timing-only based attacks where the attacker can collect training data, but only over a different connection from that against which the attack is directed. This is a significantly easier to perform attack than one which depends on training data collected over the victim link. We demonstrate that an attacker can infer the

correct web page >87% of the time when the training data is collected at a distance of up to 25 km from the victim, provided that the type of link is similar *e.g.* if the victim link uses a cable modem then the training data should be measured over a cable modem link. We also investigate the impact of distance in time between when the training data is collected and when the attack is performed and show that a success rate of >73% can be achieved with a gap of as much as 2 months between training and attack.

Chapter 6

Proportional Fair Rate Allocation for Private Shared Networks

6.1 Introduction

Timing attacks make use of the properties of the packet *stream*, rather than of individual packets, and defence against such attacks therefore requires use of traffic shaping to modify the timing pattern of the stream of packets transmitted over the network such that it becomes hard for an eavesdropper to learn about the original message. Such traffic shaping can be achieved by inserting dummy packets into the packet stream to mask idle periods, by delaying/buffering packets to modify their timing and of course by dropping packets. In practice this shaping might be performed by, for example, end hosts or by a VPN gateway.

However, such privacy enhancing traffic shaping can impose a cost on the user and on the network. For example, insertion of dummy packets increases the load on the network while buffering user packets for longer increases delay but can reduce the need for dummy packets to ensure privacy. Importantly, for users sharing a common network path this creates a joint trade-off between their privacy, throughput and delay. This is illustrated in Figure 6.1 which shows two users whose traffic is shaped to enhance privacy and is then sent over a shared link which may be subject to eavesdropping. An increase in the rate of dummy packet transmissions by user 1 reduces the available bandwidth for user 2 on the shared link, and so to maintain privacy may require user 2 to reduce the rate at which useful (non-dummy) packets are sent, to buffer packets for longer (to disrupt timing without adding extra dummy packets) or to drop more packets (again to disrupt timing patterns). Alternatively, user 2 may choose to sacrifice privacy in order to avoid reducing throughout, increasing delay *etc.* That is,

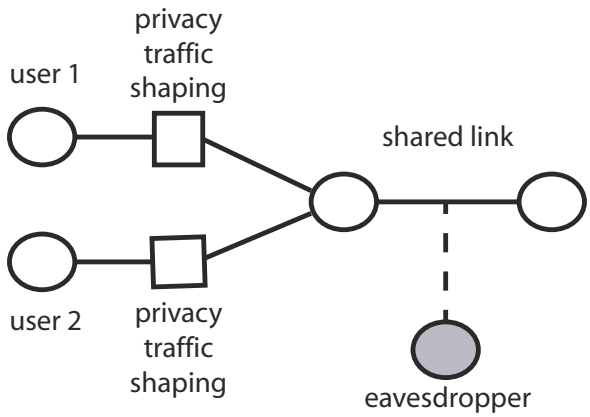


Fig. 6.1 Schematic illustrating a simple two flow example. Users apply traffic shaping to their packet streams by insertion of dummy packets to enhance their personal privacy in a shared network.

increased privacy for one user may come at the cost of decreased throughput, increased delay and/or reduce privacy for another user.

In this chapter, we initiate the study of the joint trade-off between privacy, throughput and delay in a shared network as a utility fairness problem. We derive the proportional fair rate allocation for networks of flows subject to privacy constraints and delay deadlines. To the best of our knowledge this is the first study of *fair* privacy in a shared network subject to timing attacks.

6.2 Attack Model and Privacy for a Network Flow

We consider a network path where time $t \in \{0, 1, \cdots\}$ is slotted and in slot t a new packet may arrive for transmission across the network. We let random variable $X(t) = 1$ if a packet arrives in slot t and $X(t) = 0$ otherwise. It is assumed that packets themselves contain no interesting information for an eavesdropper, which means that the packets are strongly encrypted and are of constant size. However, the sequence $\mathbf{X} := \{X(t)\}$ of packet arrivals contains information that can be used to reveal the characteristics of the traffic flow. To protect this information, the arriving packet stream is passed through a traffic shaper before being transmitted across the network. We let random variable $Y(t) = 1$ if a packet is transmitted across the path in slot t and $Y(t) = 0$ otherwise.

Our attack model is that the sequence $\mathbf{Y} := \{Y(t)\}$ of transmissions is observable by an adversary, but not the sequence $\mathbf{X}$ of arrivals. Note that in general we expect that an eavesdropper listening to network transmissions does not observe output sequence

$\mathbf{Y}$ directly, but rather only after transformation by the MAC layer *i.e.* after buffering, scheduling delay *etc.* This will generally increase privacy (this follows from the data processing inequality, and more specifically *e.g.* passing through a queue increases entropy). Nevertheless, we take a conservative approach and assume a strong attacker who can perfectly invert these changes and so recover $\mathbf{Y}$.

We use the mutual information $I(\mathbf{Y};\mathbf{X})$ between sequences $\mathbf{X}$ and $\mathbf{Y}$ as our measure of privacy, similarly to [92] and others. Mutual information measures the capacity of the information channel between sequences $\mathbf{X}$ and $\mathbf{Y}$. When $I(\mathbf{Y};\mathbf{X}) = 0$ the sequences are statistically independent and more generally mutual information captures the exposure to watermarking attacks, which can be thought of as a worst case situation in our setup, namely where the user sequence $\mathbf{X}$ has a signature intentionally embedded within its timing pattern by an adversary (*e.g.* by a web site which is being downloaded). Recall that $I(\mathbf{Y};\mathbf{X}) = H(\mathbf{Y}) - H(\mathbf{Y}|\mathbf{X})$, where $H(\mathbf{Y})$ is the entropy of the output sequence and $H(\mathbf{Y}|\mathbf{X})$ the conditional entropy between the output and input sequences. Privacy is maximised when $I(\mathbf{Y};\mathbf{X})$ is minimised (as already noted, when $I(\mathbf{Y};\mathbf{X}) = 0$, the output sequence $\mathbf{Y}$ is statistically independent from the input sequence $\mathbf{X}$ and we say transmissions are fully private).

6.3 On-Off Traffic Shaping Policy

Transmitting a packet in every slot regardless of the pattern of the arriving user traffic ensures that the transmitted packet sequence contains no information about the user traffic[1] and $I(\mathbf{Y};\mathbf{X}) = 0$. Similarly, when no packets are transmitted, $Y(t) = 0$, $t = 1, 2, \cdots$, then also trivially $I(\mathbf{Y};\mathbf{X}) = 0$. However, transmitting a packet in every slot means that when there is no information packet to send a dummy packet must be transmitted and so is clearly wasteful. And transmitting no packets at all is clearly private but not useful for communication. This motivates use of an on-off approach to transmission. That is, we specify an interval τ. In the first slot of this interval we transmit a packet (sending a dummy packet if no information packets are available[2]), and over the remaining $\tau - 1$ slots no packets are transmitted (any

[1]Formally, suppose output sequence $Y(t) = 1$, $t = 1, 2, \cdots$ *i.e.* a packet is transmitted in every slot. The entropy of the transmission at a single slot is $H(Y(t)) = p \log p + (1-p)\log(1-p)$ where $p = Prob(Y(t) = 1)$. Since $p = 1$ when a packet is always transmitted, $H(Y(t)) = 0$. The entropy of the *sequence* $\{Y(t)\}$ satisfies $H(\mathbf{Y}) \leq \sum_t H(Y(t)) = 0$ and since $H(\mathbf{Y}) \geq 0$ it follows that the entropy of the sequence $\{Y(t)\}$ is zero. Similarly, $H(\mathbf{Y}|\mathbf{X}) = 0$ and so the mutual information $I(\mathbf{Y};\mathbf{X}) = 0$.

[2]Note that it is also possible to adopt a partially private approach where, when no information packet is available to send, a dummy packet is transmitted with some probability less than one. In this case the mutual information $I(\mathbf{Y};\mathbf{X})$ will be non-zero, but can be controlled by adjusting the

arriving information packets are buffered in a queue). That is, $Y(t) = 1$ for $t = 1$ and $Y(t) = 0$ for $t \in \{2, \cdots, \tau\}$. It is known from queueing theory that deterministic service minimises the queueing delay for a specified mean service rate [45], *i.e.* periodic service minimises delay. Conveniently, periodic service also means that the pattern of packet transmissions contains no information, *i.e.* the mutual information between the transmitted sequence $\mathbf{Y}$ and the arrival process $\mathbf{X}$ is zero, $I(\mathbf{Y}; \mathbf{X}) = 0$ and so transmissions are fully private. Hence this traffic shaping approach is a minimum delay maximum privacy one. By adjusting the mean transmit rate via parameters g and τ we can tune the trade-off between the number of dummy packets sent (which waste network bandwidth) and the buffering delay experienced by information packets.

This on-off traffic shaper can be modelled as a so-called Fixed Cycle Traffic Light (FCTL), first studied in [57] in the context of vehicular traffic. Consider cars arriving at a junction controlled by a FCTL. The light has two states which divides the total cycle into fixed length green (g) and red (r) cycles. In the red cycle, a new arrival enters a queue, waiting to cross the junction. When the light turns green, cars in the queue cross the junction one at a time until the cycle lasts or queue becomes empty. Cars arriving at the junction during a green cycle and finding the queue empty proceed to cross the junction. It is this latter characteristic which makes the FCTL different from a conventional queue with periodic service. In our setup the green cycle is of duration $g = 1$ slot and the red cycle is of duraton $r = \tau - 1$ slots. The characteristics of a FCTL have been much studied, see for example the overview in [13], with estimates of the average queue length given in [57] and [72].

6.3.1 Delay

Suppose the input sequence $\{X(t)\}$ is i.i.d with $\mathbb{P}(X(t) = 1) = p$ and $\mathbb{P}(X(t) = 0) = 1 - p$. That is, the traffic arrivals form a Bernoulli process.[3] Given that the time is slotted, following [57], the average waiting time w for each packet measured in time slots is

$$w = (\tau - g)(1 - p)^{-1}\tau^{-1}(E(q_x)/p + (\tau - g + 1)/2). \tag{6.1}$$

where $E(q_x)$ denotes the length of queue right at the end of green (serving) period of the xth cycle and τ is the length of each cycle *i.e.* $\tau = r + g$.

probability with which dummy packets are sent. However, we leave this more general case for future work.

[3]This is one of the common assumptions in traffic generation models. The investigation of arbitrary arrivals is left for future work.

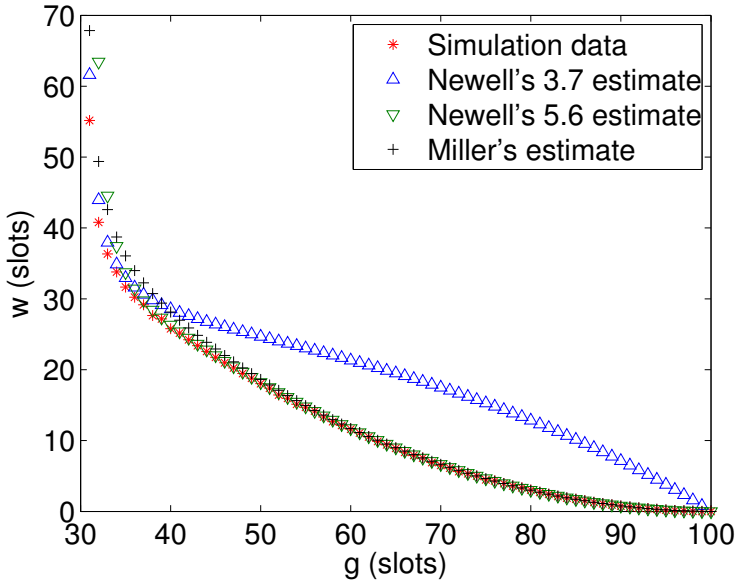


Fig. 6.2 Comparison of Newell's (equations (3.7) and (5.6) in [57]) and Miller's estimates for the waiting time at a FCTL with simulation data. The simulation values are averaged over 1000 cycles with $p = 0.3$ and $\tau = 100$. The data for $g/\tau < p$ is excluded as the queue is unstable in this region.

In order to calculate w we need to find the equilibrium distribution for q_x and evaluate $E(q_x)$. While the calculation of q_1 and q_2 is straightforward, the evaluation of q_3 onwards quickly becomes tedious and the expressions cumbersome. Several attempts have therefore been made to derive accurate estimates of $E(q_x)$ which are simpler in nature. Two estimates, that of Newell [57] and of Miller [72], are illustrated in Figure 6.2 and compared against numerical simulation data. In the sequel we use Miller's estimate due to its accuracy and simplicity. This is given by

$$E(q_x) \approx \max\{\frac{2p\tau - g}{2(g - p\tau)}.(1 - p), 0\} \tag{6.2}$$

6.3.2 Rate of Dummy Packet Transmissions

In addition to the delay introduced by traffic shaping we are also interested in the fraction d of slots expended on dummy packet transmissions (when no information packet is available to send at a slot in an on cycle). The information packet arrivals ν

during an off period are distributed as:

$$Prob(\nu = k) = \binom{T}{k} p^k (1-p)^{T-k}, \tag{6.3}$$

and the associated probability generating function is $K(z) = [(1-p)+pz]^T$. Regarding $\Pi(z)$, the probability generating function for the limiting distribution π_j the length of the queue at the start of the first slot of an on period, we have:

$$\Pi(z) = \frac{\sum_{j=0}^{g-1} \pi_j (z^g - z^j)}{z^g/K(z) - 1} = \frac{\sum_{j=0}^{g} \pi_j (z^g - z^j)}{z^g[(1-p)+pz]^{-T} - 1}. \tag{6.4}$$

We know $\Pi(1) = 1$, which means after using L'hopital's rule

$$d = \frac{1}{\tau} \sum_{j=0}^{g-1} \pi_j (g - j) = \frac{g}{\tau} - p, \tag{6.5}$$

the average number of dummy packets transmitted over cycle.

We will assume that $g - p\tau > 0$. This means that on average our service can accommodate all of the arrivals to the queue and the queue is stable. Stability can be seen by looking at the polynomial $z^g[(1-p)+pz]^{-T} - 1$ which, for stability, should have no zeros in or on the unit circle. Now,

$$z^g = [(1-p)+pz]^\tau, \; gz^g = p\tau[(1-p)+pz]^{\tau-1} \tag{6.6}$$

Dividing the two equations, for $|z| < 1$ we should have $p\tau > g$, but this is excluded when $g - p\tau > 0$.

6.3.3 Example

In order to investigate the effectiveness of on-off traffic shaping on privacy, we conducted the following test: we collected packet timestamp traces from 10 different web sites. The traces are of approximately same length. We compared all of the samples with each other using Dynamic Time Warping and calculated their F-distance, see Chapter 4.3. The F-distance is smaller when traces are similar and increases as they become more different. It can be seen in Figure 6.3 that the average distance between the traces for different web sites is relatively large, which would enable an attacker to distinguish between them.

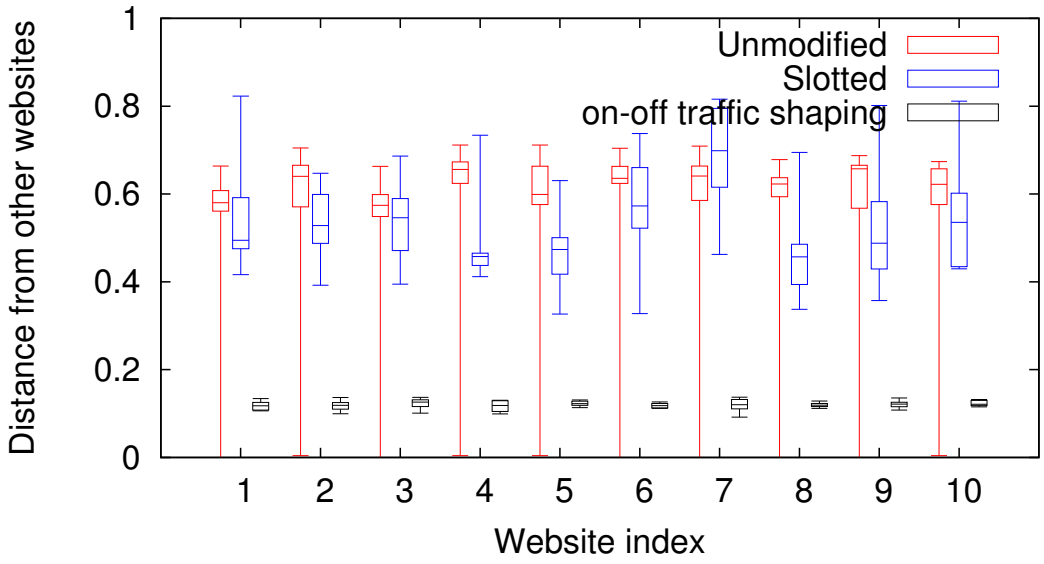


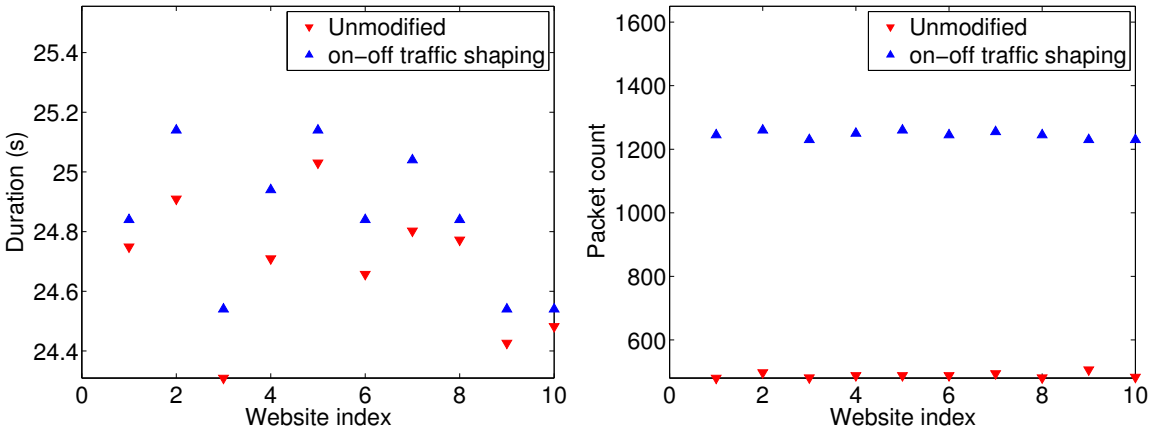
Fig. 6.3 Comparison of F-distance between different web traces for unmodified, slotted and queued+slotted scenarios. The slot size is 0.01s, $g = 5$, $\tau = 10$ and the window size for evaluating F-distance is 0.2.

We then manually divided time into slots (of 10ms duration) and adjusted the packet timings so that they are only sent at the beginning of a slot[4]. It can be seen in Figure 6.3 that time slotting decreases the average distance between packet traces, the decrease is relatively small and would still allow an attacker to distinguish between different the traces for different web sites.

Finally we applied on-off traffic shaping and calculated the F-distance. It can be seen that there is now a considerable drop in the average distance, and also in the variance. This indicates that the modified traces are now much more similar, making it difficult for the attacker to distinguish between web sites.

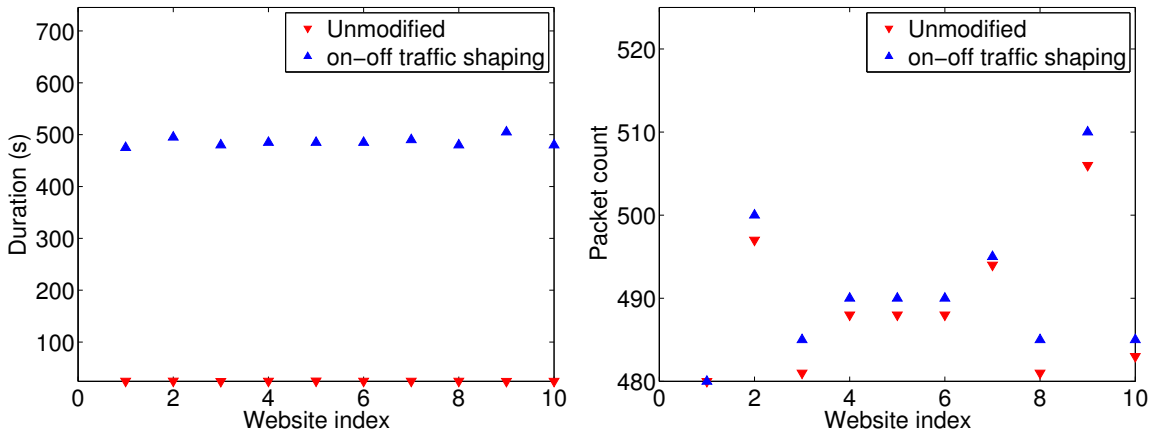
As already noted, traffic shaping introduces a queueing delay and the transmission of additional dummy packets. Figure 6.4 shows the measured delay and number of dummy packets when $g/\tau = 0.5$. It can be seen that the delay is negligible since τ is small but the number of dummy packets is almost 2 times that of the real traffic. As g/τ is decreases, the number of dummy packets transmitted falls, but the delay increases. This is illustrated in Figure 6.5 for $g/\tau = 0.01$.

[4]Note that using an actual time slotted tunnel adds an additional distortion due to network protocols. So by manually time slotting the time we are considering a slightly easier scenario for the attacker.



(a) Duration (b) Packet count

Fig. 6.4 Comparison between transmission duration and packet count for unmodified and on-off shaped web traces. $g/\tau = 0.5$.



(a) Duration (b) Packet count

Fig. 6.5 Comparison between transmission duration and packet count for unmodified and on-off shaped web traces. $g/\tau = 0.01$.

6.4 Proportional Fair Privacy

We consider a shared network where each user applies traffic shaping before transmitting their traffic over the network, *e.g.* see Figure 6.1. Note that traffic shaping is applied individually to each user's traffic, which is required to prevent active traffic injection attacks (where an adversary sharing a queue with a user injects traffic in a way that reveals the presence/absence of a user packet in the queue [42]). In practice this setup might correspond to a VPN where the client shapes and encrypts arriving traffic before transmitting it across the Internet to a VPN gateway. As already noted, buffering and shaping may introduce delay (via buffering) and consume bandwidth (via dummy packet transmissions) and there is a joint trade-off between privacy, delay and throughput for the users sharing the network.

6.4.1 Network Model

We consider a multi-user network serving a set of $\mathcal{N}$ users and a set $\mathcal{F}$ of flows. Each flow $f \in \mathcal{F}$ has a source $s_f \in \mathcal{N}$, a mean offered load rate μ_f and a mean delay deadline σ_f. The network uses a scheduler that can service the offered load provided the aggregate flow usage satisfies

$$\sum_{f \in \mathcal{F}} \frac{\mu_f}{\psi_f} \leq 1 \tag{6.7}$$

where ψ_f is the physical transmit rate for flow f and so μ_f/ψ_f is the mean fraction of airtime used by flow f and for arrival rate p_f and dummy rate d_f we have $\mu_f = p_f + d_f$.

6.4.2 Convexity of Traffic Shaping Delay

As already noted, on-off traffic shaping introduces additional delay. When on-time $g = 1$ we have

$$E(q_x) \approx \max\{\frac{(2p - c)(1 - p)}{2(c - p)}, 0\} \tag{6.8}$$

and mean waiting time

$$w = \frac{1 - c}{1 - p}\left[\frac{E(q_x)}{p} + \frac{1}{2c}\right] \tag{6.9}$$

where $c = g/\tau > p$.

Lemma 1 (Convexity of Delay)**.** The mean waiting time w in (6.9) is convex in the arrival rate p and dummy rate $d = c - p$, although not jointly convex, for $p \in [0, 1]$, $c \in (p, 1]$.

Proof. When $c > 2p$ (so $E(q_x) = 0$), the second derivatives of w with respect to p and c are

$$w_{pp} = \frac{1-c}{c(1-p)^3},\ w_{cc} = \frac{1}{c^3(1-p)}\, w_{pc} = -\frac{1}{2c^2(1-p)2}$$

It can be seen that $w_{pp} > 0$ and $w_{cc} > 0$ since $p, c \in [0, 1]$. When $c < 2p$ (so $E(q_x) > 0$), we have

$$w_{pp} = (1-c)\left[\frac{1}{(c-p)^3} - \frac{1}{p^3} + \frac{1}{c(1-p)^3}\right] \tag{6.10}$$

$$w_{cc} = \frac{1-p}{(c-p)^3} + \frac{1}{c^3(1-p)} \tag{6.11}$$

$$w_{pc} = -\frac{1}{2}\left[\frac{2-p-c}{(c-p)^3} + \frac{1}{p^2} + \frac{1}{c^2(1-p)^2}\right] \tag{6.12}$$

Since $c > p$ then $w_{cc} > 0$. Since $c < 2p$, $w_{pp} > 0$. However the Hessian need not be positive semidefinite (for example when $p = 0.5$ and $c = 0.9$). Since $w_{pp} > 0$ and $w_{cc} > 0$, the delay w is convex in p and c individually, but since the Hessian is not positive semidefinite w is not jointly convex in these variables. Now $d = c - p$ is a linear function of c and p. Convexity is preserved under linear transformations and so the stated result follows. □

Note that since the on-time g and cycle time τ are expressed as numbers of slots, they are integer valued $g, \tau \in \mathbb{N}$ and so the domain of ratio c is the rational numbers $\mathbb{Q}$ rather than the real-valued numbers. We therefore consider the relaxed problem where c takes values in $(p, 1] \subset \mathbb{R}$ in order to ensure convexity. This relaxation does not, however, entail any loss of generality. Rather than using fixed on-time $g \in \mathbb{N}$, define a sequence $g_k \in \mathbb{N}$, $k = 1, 2, \cdots$. For a given value of $c \in \mathbb{R}$, by queue continuity [43] an integer-valued sequence g_k exists such $|\sum_k (g_k - \hat{g})|$ is bounded and so the waiting time can be made arbitrarily close to that with specified $\hat{g} = c\tau \in \mathbb{R}$.

6.4.3 Proportional Fair Allocation

We consider the following optimisation P which maximises log-rate (and so is proportional fair) subject to network and traffic shaping delay constraints:

$$\min_{\mathbf{p},\mathbf{d}} \quad U(\mathbf{p}) := \sum_{f\in\mathcal{F}} -\log(p_f) \tag{6.13}$$

$$\text{s.t.} \quad \mathbf{w} \preceq \boldsymbol{\sigma} \tag{6.14}$$

$$\sum_{f\in\mathcal{F}} \frac{p_f + d_f}{\psi_f} \leq 1 \tag{6.15}$$

$$\mathbf{p}, \mathbf{d} \in [0,1]^{|\mathcal{F}|} \tag{6.16}$$

with

$$w_f = \frac{1-(p_f+d_f)}{2(1-p_f)}\Big[\max\{\frac{(p_f-d_f)(1-p_f)}{p_f d_f}, 0\} + \frac{1}{(p_f+d_f)}\Big] \tag{6.17}$$

for all flows $f \in \mathcal{F}$ and where $\mathbf{w} = [w_f]$, $\boldsymbol{\sigma} = [\sigma_f]$, $\mathbf{p} = [p_f]$, $\mathbf{d} = [d_f]$ for $f \in \mathcal{F}$ are vectors in $\mathbb{R}^{|\mathcal{F}|}$. Constraints (6.14) impose the requirement that flow delay deadlines are met, while (6.15) ensures that the flow rates (including both information and dummy transmissions) can be scheduled by the network.

6.4.4 Solving Non-Convex Optimisation P

Optimisation P is not convex because, by Lemma 1, the delay constraints are not jointly convex in $\mathbf{p}$ and $\mathbf{d}$. Nevertheless, these constraints are convex in $\mathbf{p}$ and $\mathbf{d}$ individually. This suggests the use of an alternating approach to solve P. Namely, solve for $\mathbf{p}$ holding $\mathbf{d}$ constant, then solve for $\mathbf{d}$ holding $\mathbf{p}$ constant. Let $\mathbf{p}_k^*$, $\mathbf{d}_k^*$, $k = 1, 2, \cdots$ denote the sequence of alternating solutions found in this way. Each solution is feasible for problem P. Further, since each individual optimisation is convex, we can find a global minimum and so $U(\mathbf{p}_{k+1}^*) \leq U(\mathbf{p}_k^*)$. Hence, the sequence $\mathbf{p}_k^*$, $\mathbf{d}_k^*$ is guaranteed to converge to a feasible stationary point of problem P. While this stationary point is, in general, sub-optimal, in practice we have found that it is usually close to a global optimum.

To carry out each optimisation we use a subgradient method for simplicity and because of its suitability for distributed implementation. Of course other methods might also be used. The resulting procedure is summarised as follows:

Algorithm 1 Alternating Solution Method

iterate s**:**
  iterate t**:**
    $\mathbf{p}(t+1) = \mathbf{p}(t) - \alpha\partial_{\mathbf{p}} L_{\mathbf{p}}(\mathbf{p}(t), \mathbf{d}(s-1), \boldsymbol{\lambda}_p(t))$
    $\boldsymbol{\lambda}_p(t+1) = [\boldsymbol{\lambda}_p(t) + \alpha\partial_{\boldsymbol{\lambda}} L_{\mathbf{p}}(\mathbf{p}(t), \mathbf{d}(s-1), \boldsymbol{\lambda}_p(t))]^+$
  loop
  iterate t**:**
    $\mathbf{d}(t+1) = \mathbf{d}(t) - \alpha\partial_{\mathbf{d}} L_{\mathbf{d}}(\mathbf{p}(s), \mathbf{d}(t), \boldsymbol{\lambda}_d(t))$
    $\boldsymbol{\lambda}_d(t+1) = [\boldsymbol{\lambda}_d(t) + \alpha\partial_{\boldsymbol{\lambda}} L_{\mathbf{d}}(\mathbf{p}(s), \mathbf{d}(t), \boldsymbol{\lambda}_d(t))]^+$
  loop
loop

with Lagrangians,

$$L_{\mathbf{p}}(\mathbf{p}, \mathbf{d}^*, \boldsymbol{\lambda}) = \sum_{f\in\mathcal{F}} -\log(p_f) + \sum_{f\in\mathcal{F}} \lambda_{1,f}(w_f - \sigma_f) + \lambda_2((\sum_{f\in\mathcal{F}} p_f + d_f^*) - 1) + \sum_{f\in\mathcal{F}} \lambda_{3,f}(p_f - 1) - \sum_{f\in\mathcal{F}} \lambda_{4,f} p_f$$

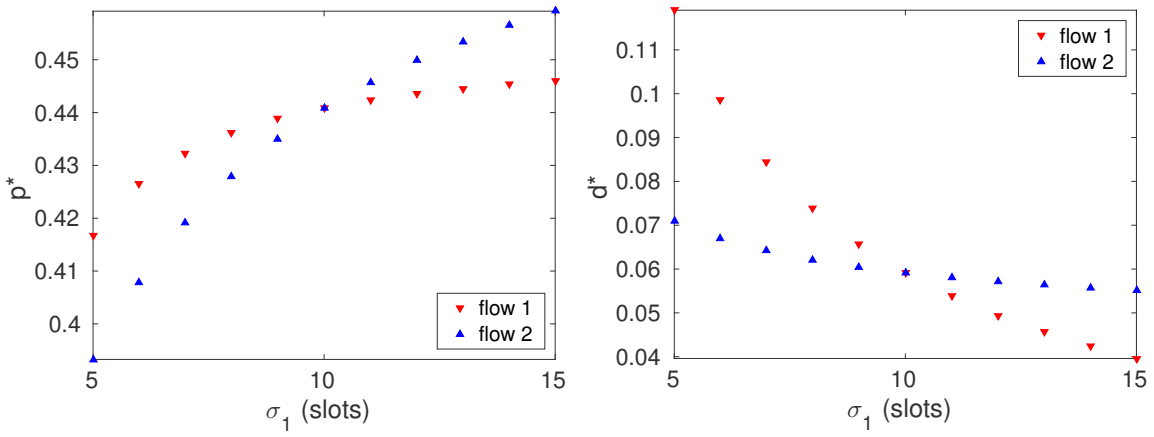
$$L_{\mathbf{d}}(\mathbf{p}^*, \mathbf{d}, \boldsymbol{\lambda}) = \sum_{f\in\mathcal{F}} \lambda_{1,f}(w_f - \sigma_f) + \lambda_2((\sum_{f\in\mathcal{F}} p_f^* + d_f) - 1) - \sum_{f\in\mathcal{F}} \lambda_{5,f} d_f$$

6.4.5 Examples

We present a number of examples to illustrate the proportional fair allocation in a fully private shared network.

Example 1: Fully private network. Consider a network with two users having different delay deadlines. We let delay deadline $\sigma_2 = 10$ and vary σ_1 between $\{\sigma_2 - 5, \sigma_2 + 5\}$ to observe the impact of the delay deadline on users' network share in a fully private network. Figure 6.6 shows the proportional fair p^* and d^* vs σ_1. It can be seen that users' throughput and dummy rate are proportional to their delay deadlines *i.e.* users with lower deadline are allowed to transmit more real and dummy traffic resulting a larger network share.

Example 2: Mix of private and non-private flows. As already noted, users in a shared network need to sacrifice throughput and/or delay to achieve full privacy.



(a) Optimal throughputs (b) Optimal dummy rates

Fig. 6.6 Illustrating optimal throughput and dummy rate for delay deadlines $\sigma_2 = 10$ and $\sigma_1 = \{5, \ldots, 15\}$.

However, staying within our traffic shaping framework, a user can choose to ignore privacy by sending no dummy packets and using the full cycle length for transmitting information packets. Of course this allows an adversary to see their packet arrivals. The optimization problem for this scenario is similar to 6.13 except that the delay and dummy rate constraints are now

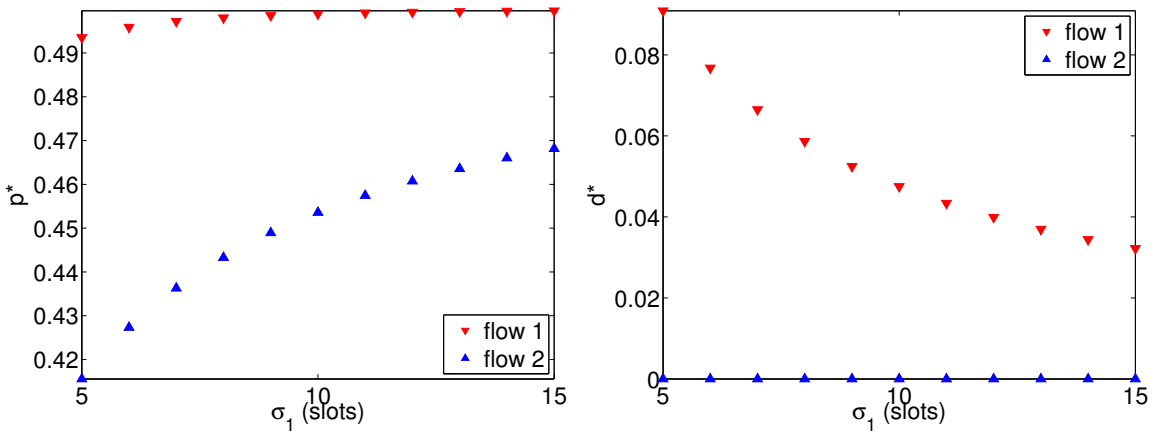
$$\begin{aligned} w_f \leq \sigma_f, \quad & f \in \mathcal{F}_{private} \\ d_f > 0, \quad & f \in \mathcal{F}_{private} \\ d_f = 0, \quad & f \in \mathcal{F} - \mathcal{F}_{private} \end{aligned}$$

where $\mathcal{F}_{private} \subset \mathcal{F}$ is the set of private users.

In this example we consider similar conditions to those in Example 1, but now User 2 is non-private. Figure 6.7 shows p^* and d^* vs the delay deadline of User 1. It can be seen, that User 1 consistently gets higher throughput than in a fully private network (compare Figures 6.7a and 6.6a).

6.5 Summary and Conclusions

In this chapter, we introduce a rate allocation scheme for private shared networks. First, a defence is proposed against timing only traffic analysis attacks which protects the user by transforming their packet arrival time sequence into one which contains no information about the packet arrival pattern of the original sequence. The transforma-



(a) Optimal throughputs (b) Optimal dummy rates

Fig. 6.7 Illustrating optimal throughput and dummy rate for a mix of private and non-private users. User 1 is private with $\sigma_1 = \{5, \ldots, 15\}$ while User 2 ignores privacy by transmitting no dummy packets and using the full cycle length for transmitting information packets.

tion however imposes a delay on transmission and consumes bandwidth by transmitting dummy traffic. We address a shared network scenario where the performance of one user can affect the network experience of another. This leads to a further analysis of the resulting trade-off between user privacy and quality of experience and to the design of a proportional fair rate allocation algorithm.

Chapter 7

Privacy-Enhanced VPN Resistant to Timing-Analysis

7.1 Introduction

In this chapter we consider extending an encrypted tunnel to provide defence against timing attacks. Unlike the previous chapter, this admits general traffic arrivals. The basic idea is to ensure privacy by serving the incoming traffic using predefined traffic patterns, called "traces". The service rate is controlled by activating sufficient number of traces to match the rate of arrivals. The delay, throughput and privacy performance achieved is evaluated using and prototype implementation of a privacy-enhanced VPN.

7.2 Achieving Privacy

7.2.1 Class of Attacks Considered

We consider an attacker of the type illustrated in Figure 7.1. The attacker can sniff packets traversing the encrypted tunnel and the attacker's objective is to use this information to guess, with high probability of success, the web pages which the client visits. Our main concern is with the information leaked to the attacker by the timing of packet transmissions. This is because 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 [8]. In contrast, powerful timing-based attacks have recently been demonstrated that are potentially a practically important class of attack against current and future VPNs.

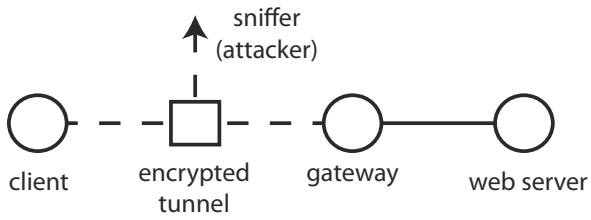


Fig. 7.1 Schematic illustrating attacker of the type considered. A client machine is connected to an external network via an encrypted tunnel (ssh, SSL, IPSec etc.). The attacker can sniff packets traversing the tunnel, but has no other information about the clients activity.

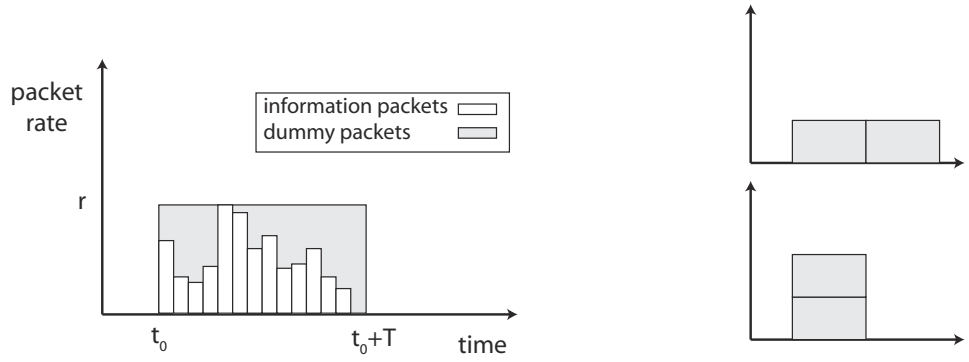
Note that we do not seek to address attacks that target the end host itself (viruses *etc.*), nor attacks based on active packet injection by the attacker (*e.g.* to force packet loss so as to manipulate user or web server behaviour).

7.2.2 Indistinguishability

We are not concerned with concealing the fact that the user is browsing the web but rather with concealing the particular pages visited. Therefore, given an observed packet sequence on the network we would like there to be a sufficiently large number of different web fetches, or combinations of web fetches, that could generate that packet sequence with reasonable probability. When this holds then a user can reasonably deny that they have fetched any particular page, claiming that instead they fetched a different page or combination of pages. Privacy is therefore embodied in the indistinguishability of sequences of web fetches given with regard to the packet sequences which they generate.

It is important to note that we do not insist that fetches of single web pages individually be indistinguishable. Rather, we require that the observed packet sequence corresponding to the fetch of each web page can be reasonably explained by other combinations of web fetches. So, for example, the packet sequence generated by the fetch of a large web page could equally have been generated by sequences of fetches of small web pages. Fetches of single web pages must still be indistinguishable from one another when they cannot plausibly be generated by combinations of other page fetches. That is, we require fetches of sufficiently small web pages to atomic in the sense that they are indistinguishable from one another.

There is also a need to ensure diversity, in an appropriate sense, amongst the sequences of web fetches that can explain an observed packet sequence. For example, if only combinations of fetches of pages from the same web site could generate an ob-



(a) Uniform profile transformation (b) Consecutive and stacked traces

Fig. 7.2 Illustrating use of dummy packets to force the number and timing of transmitted packets to conform to an uninformative pattern or "trace". When insufficient information packets are available, dummy packets are inserted as needed. When too many information packets are available they are buffered until a transmission opportunity becomes available.

served packet sequence then privacy is undermined. However, given the huge diversity of web pages in the Internet we do not expect this to be a major concern.

7.2.3 Defence By Use of "Traces"

The basic problem with the packet sequence generated by a web fetch is that it contains many packets (often several hundred, frequently more) and so amounts to an observation of a point in high dimensional space. It is increasingly recognised that high dimensional data carries a major risk of de-anonymisation since data points tend to be sparsely distributed in high dimensions (each point can individually distinguished). Our approach is therefore to reduce the dimension of the observed packet sequence data. We do this by gathering packets into larger groups and transmitting these groups in a uniform way.

This approach is illustrated in Figure 7.2a. Here, a web fetch starts at time t_0. Dummy packets (indicated in grey) are inserted so that the observed sequence of packets, which is combination of dummy plus user packets, has a uniform profile and duration. We refer to this uniform profile as a "trace". The rate and duration of the trace are design parameters, that we will return to later[1]. In this example the web fetch completes by time $t_0 + T$ and so fits within a single trace. However, if the

[1]We might also use a trace where the rate is not constant over time and also allow the trace used to be drawn from a pre-defined set of traces.

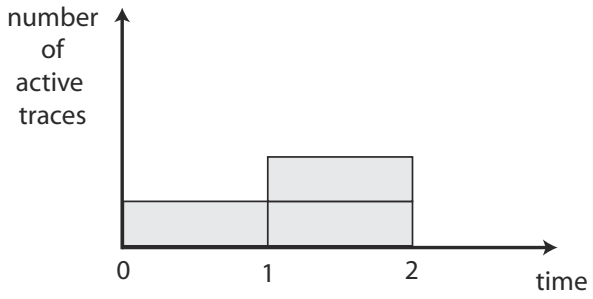


Fig. 7.3 Example illustrating an observed sequence of traces.

duration exceeded T then a second trace would be started so as to mask the additional user packets, see upper schematic in Figure 7.2b. Similarly, if the rate at which user packets arrive much exceeds the rate of a trace then two traces can be started in parallel, see lower schematic in Figure 7.2b. With this approach the observed packet sequence now consists of a sequence of traces.

Given an observed sequence of such traces, the indistinguishability question becomes: how many different sequences of web fetches can reasonably generate the observed trace sequence. For example, consider the sequence of traces shown in Figure 7.3. Time is slotted, with each slot corresponding to the duration of a trace. Traces start at the beginning of a slot, and suppose a maximum of n traces are possible in each slot (*e.g.* limited by the link capacity). Suppose we have a set W of web pages of interest. Of these, fetches for web pages $W_1 \subset W$ can be covered by a one trace, $W_2 \subset W$ by two traces and so on. Let p_w be the probability that web page $w \in W$ is fetched and assume, for simplicity, that pages are fetched independently (this is clearly not true in practice of course). Let X_w be a random variable which takes value 1 when web page w is fetched and 0 otherwise. Let $T \subset \{0, n\}^k$ denote the observed sequence of traces of length k slots, with $T = (1, 2, 0, \ldots, 0)$ in Figure 7.3 . Then,

$$\mathrm{P}(X_w = 1 | T, w \in W_1) = 1 - \prod_{j=1}^{k} \left(1 - \frac{p_w}{\sum_{i \in W_1} p_i}\right)^{T_j} \tag{7.1}$$

and similarly we can calculate $\mathrm{P}(X_w = 1 | T, w \in W_2)$ etc. In the example in Figure 7.3 suppose $p_w = 1/|W|$. Then for a web page $w \in W_1$ that fits inside a single trace the probability that it was fetched is $1 - \left(1 - \frac{1}{|W_1|}\right)\left(1 - \frac{1}{|W_1|}\right)^2$. For example, when $|W_1| = 100$ this probability is 0.03 and when $|W_1| = 1000$ the probability is 0.003 and it can be seen that as provided the cardinality of W_1 is reasonably large a user can with high plausibility deny that they fetched a given web page.

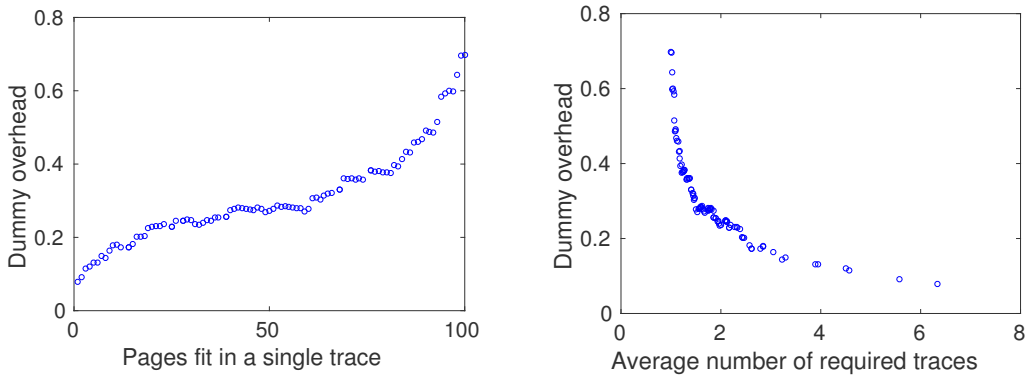


Fig. 7.4 Illustrating the overhead of dummy packets vs trace duration. Data is for alexa top 100 web pages.

Observe in this example that the size of the sets W_1, W_2 *etc.* is important and, in particular, that set W_1 needs to be sufficiently large. That is, as already noted, a trace has to be of sufficient rate and duration that many small web sites fit inside a single trace. Also, the larger the rate and duration of a trace the greater the level of indistinguishability achieved.

Note that in the above example an attacker can infer that the user did not fetch any web pages that cannot fit inside three traces or fewer. However, given the extremely large number of possible web pages and the fact that most of these pages are not fetched by a given user we argue that the absence of a fetch for a web site is much less informative to an attacker than the presence of a fetch.

We note also that the option exists to inject dummy traces into the tunnel link to add a further level of deniability. Using such an approach the potential exists to formulate indistinguishability as a form of differential privacy. Namely, such that the addition of the fetch of any individual web page has limited impact on the sequence of traces observed on the link. However, we leave this as future work and do not pursue it further here.

7.2.4 Rate and Duration of a Trace

By buffering user packets at tunnel ingress until enough are available to fill a trace, we can avoid the overhead of sending dummy packets. However, this comes at the cost if increased delay, perhaps much increased delay. However, in modern networks, at least in the western world, quality of service is typically achieved over-provisioning of network capacity and as a result bandwidth is often relatively plentiful. Conversely,

users are known to be sensitive to delay when using online services. For example, Amazon estimates that a 100ms increase in delay reduces its revenue by 1% [24], Google measured a 0.74% drop in web searches when delay was artificially increased by 400ms [98] while Bing saw a 1.2% reduction in per-user revenue when the service delay was increased by 500ms [66]. Hence, in order to enhance resistance to traffic analysis attacks it is preferable to sacrifice some bandwidth by sending dummy packets rather than incur excessive delay.

To gain some insight into the overhead of dummy packets associated with different durations of trace we fetched the home pages from the alexa top 100 finance and health web sites in Ireland. Figure 7.4a plots the average fraction of packets in a trace which are dummy packets vs the number of web sites that fit inside the trace. A trace of duration 10890ms is needed to cover all 100 web sites and a trace of duration 4947ms to cover 50 web sites. As might be expected, the fraction of dummy packets increases as the duration of the trace is increased to include more web pages. When all 100 web pages fit inside a single trace the overhead is around 70% but this fall to around 30% for a trace that covers 50 web pages. Figure 7.4b plots the fraction of dummy packets vs the maximum number of consecutive traces used to cover all 100 web pages (so the value for one trace corresponds to the data in Figure 7.4a). It can be seen that as the number of traces used increases the overhead falls. However, use of smaller traces reduces the level of indistinguisability provided and so a trade-off exists between privacy and dummy packet overhead.

Since it is hard to analytically quantify the trade-off between privacy, dummy packet overhead and delay (the delay aspect is especially difficult to analyse mathematically), we will revisit this trade-off shortly using experimental data.

7.3 Scheduling Traces

A privacy-enhanced tunnel using the trace approach must schedule the start-up of traces so as to minimise user delay and dummy packet overhead. The design and analysis of an efficient scheduler for this is the main technical challenge that needs to be solved in order to construct such a tunnel and is addressed in this section.

7.3.1 Network Setup

The setup considered is illustrated in Fig 7.5. Let $\mathcal{U} = \{1, 2, \cdots, n_u\}$ denote the set of users, $a_k^{(u)} \in \mathbb{N}$ denote the number of packet arrivals for user u at time k and

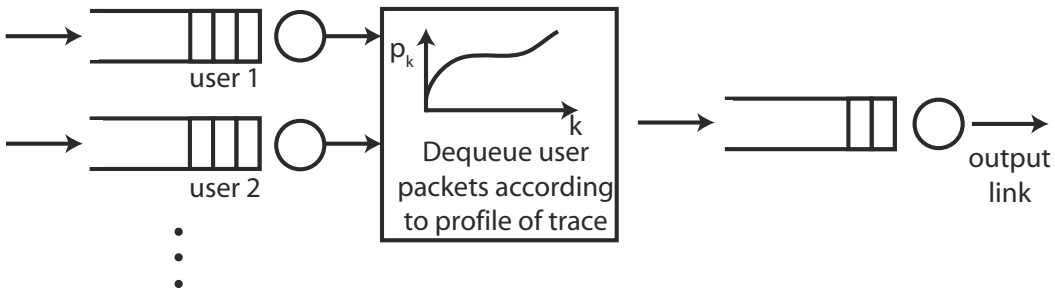


Fig. 7.5 Schematic illustrating the VPN gateway setup considered.

$\bar{a}^{(u)} := \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} a_k^{(u)}$ the average number of packet arrivals. Packets for each user u are held in separate queues, with the queue occupancy for user u at time k being denoted by $q_k^{(u)}$. Letting $w_k^{(u)} \in \mathbb{N}$ be the number of packets dequeued at time k then $q_{k+1}^{(u)} = [q_k^{(u)} + a_k^{(u)} - w_k^{(u)}]^+$ where the notation $[\cdot]^+$ has the usual meaning.

User packets are dequeued according to a predefined *trace*. Multiple traces may be active simultaneously in order to increase the rate at which user packets are dequeued. A trace f is a sequence $p_j^{(f)} \in \mathbb{N}$, $j = 1, 2, \ldots, n^{(f)}$ where $p_j^{(f)}$ is the number of packets to be transmitted at time j and $n^{(f)}$ is the duration of the trace. Importantly, when no user packets are available to send at time j, then dummy packets will be transmitted to ensure that $p_j^{(f)}$ packets are always transmitted. Let $P^{(f)} = \sum_{j=1}^{n^{(f)}} p_j^{(f)}$ denote the total number of packets in trace f. In general we might have a family of m different traces $\mathcal{F}$ to choose from.

Packets from a trace are then queued at the output link before transmission over the tunnel, see Fig 7.5. As is usual in Internet links we leave rate control to the end hosts. If the aggregate send rate persistently exceeds the output link capacity then the queue at this link will eventually overflow and cause packet loss which end hosts can then use as a congestion indicator, *e.g.* by using TCP congestion control.

New traces from family $\mathcal{F}$ are started as needed to service user packet arrivals, and multiple copies of a trace may be active at the same time so as to increase the sending rate if needed. Indexing these active traces by $1, 2, \cdots$ let $\mathcal{T}$ denote the set of trace indices, τ_t the start time of trace $t \in \mathcal{T}$ and f_t the member of family $\mathcal{F}$ to which the trace corresponds. At time k the set of active traces is $\mathcal{T}_k := \{t \in \mathcal{T} : k \in \{\tau_t, \ldots, \tau_t + n^{(t)}\}\}$ and the number of packets that are transmitted at time k is $\sum_{t\in\mathcal{T}_k} p_{k-\tau_t}^{(f_t)}$, therefore $\sum_{u\in\mathcal{U}} w_k^{(u)} \leq \sum_{t\in\mathcal{T}_k} p_{k-\tau_t}$. The average rate of packet transmissions can be no more than the capacity c in packets/slot of the outgoing link *i.e.* $\lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \sum_{t\in\mathcal{T}_k} p_{k-\tau_t}^{(f_t)} < c$.

Since $\sum_{t\in\mathcal{T}_k} p_{k-\tau_t}^{(f_t)}$ packets are sent then if there are insufficient user packets available to send at time k we need to send $\sum_{t\in\mathcal{T}_k} p_{k-\tau_t}^{(f_t)} - \sum_{u\in\mathcal{U}} w_k^{(u)}$ dummy packets. Our task

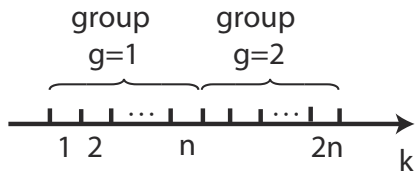


Fig. 7.6 Gathering slots into groups of n consecutive slots. In the synchronised case a trace can only be activated in the first slot of a group, and will have completed by the last slot in that same group since traces are of duration n slots.

is to activate traces and thereby adjust the number of user packets transmitted $w_k^{(u)}$ so as to minimise the number of dummy packets sent while still servicing all of the user packets in a timely manner.

To simplify the presentation that follows we will confine ourselves to setup with a single type of trace, *i.e.* family $\mathcal{F}$ contains a single member, the extension to multiple types of trace being straightforward. We take advantage of this to streamline notation, shortening $p_k^{(f_t)}$ to p_k and $n^{(t)}$ to n.

7.3.2 Synchronised Scheduler

To proceed we first construct an approximation to the above problem setup where active traces start/stop in a synchronised fashion. Traces started in slot 1 finish at slot n (since traces are of n slots duration), traces started at slot $n+1$ finish at slot $2n$ and so on. Hence, by restricting ourselves to starting traces at times $1, n+1, 2n+1, \ldots$ we can ensure synchronised start/stop of traces and avoid partial overlapping of traces. We will relax this restriction later, but the absence of overlapping traces simplifies the initial analysis and assists with gaining insight into the design issues to be considered.

Formally, partition time slots $k = 1, 2, \ldots$ into groups of n slots, see Fig 7.6. Denote the first group of slots by $\mathcal{G}_1 := \{1, \ldots, n\}$, the second by $\mathcal{G}_2 := \{n+1, \ldots, 2n\}$ and so on. Suppose a new trace can only be activated at the start of a group of slots. Since each group is n slots long each trace will therefore have finished by the end of the group of slots in which it started. Letting y_g denote the number of traces active in group g, since the traces within a group are all activated at the same time the number of packets that are transmitted in slot $k \in \mathcal{G}_g$ simplifies to $y_g p_{k-(g-1)n}$. The capacity constraint on the outgoing link therefore now becomes $\lim_{K\to\infty} \frac{1}{K} \sum_{g=1}^{K} y_g \frac{P}{n} < c$ *i.e.* $\bar{y}P < \bar{c}$ where $\bar{y} := \lim_{K\to\infty} \frac{1}{K} \sum_{g=1}^{K} y_g$ is the average number of traces active at the same time and $\bar{c} := cn$.

Let $b_g^{(u)} := \sum_{k\in\mathcal{G}_g} a_k^{(u)}$ be the number of packet arrivals for user u in group of slots g and $\bar{b}^{(u)} := \lim_{K\to\infty} \frac{1}{K} \sum_{g=1}^{K} b_g^{(u)} = \bar{a}^{(u)} n$ be the average number of packet arrivals for

user u. Then provided the average number of packets served $\lim_{K\to\infty} \frac{1}{K} \sum_{g=1}^{K} \sum_{k\in\mathcal{G}_g} w_k^{(u)} = \bar{y}P > \sum_{u\in\mathcal{U}} \bar{b}^{(u)}$ then the arriving user packets can all be served.

Convex Optimisation

To minimise the number of dummy packets sent while serving the user packet arrivals we end up with the following formal convex optimisation P,

$$\begin{aligned} &\min_{0\leq\bar{y}\leq y_{max}} \gamma\bar{y} \\ s.t. \quad &\sum_{u\in\mathcal{U}} \bar{b}^{(u)} \leq \bar{y}P \end{aligned}$$

where $\gamma \geq 0$. This optimisation is formal in the sense that we can see that the solution is trivially $\bar{y}^* = \frac{1}{P}\sum_{u\in\mathcal{U}} \bar{b}^{(u)}$ provided $\frac{1}{P}\sum_{u\in\mathcal{U}} \bar{b}^{(u)} \leq y_{max}$. However, we would like to derive an algorithm that finds the solution without requiring a priori knowledge of the mean user packet arrival rates $\bar{b}^{(u)}$ and we would also like to ensure the stability of the queue in the scheduler, and it is for these reasons that the formulation as an optimisation will prove useful.

Online Solution

The Lagrangian for the above optimisation is $L(\bar{y}, \lambda) := \gamma\bar{y} + \lambda\left(\sum_{u\in\mathcal{U}} \bar{b}^{(u)} - \bar{y}P\right)$. By the KKT conditions it can be seen that the multipliers satisfies $\lambda - \gamma/P = 0$ at an optimum. Suppose the Slater condition is satisfied and so strong duality holds. Then a solution can be found using the following dual subgradient update

$$x_g \in \arg\min_{x\in[0,y_{max}]} \partial L(\bar{y}_g, \lambda_g)x \tag{7.2}$$

$$= \arg\min_{x\in[0,y_{max}]} (\gamma - \lambda_g P)x \tag{7.3}$$

$$\bar{y}_{g+1} = (1-\beta)\bar{y}_g + \beta x_g \tag{7.4}$$

$$\lambda_{g+1} = [\lambda_g + \alpha(\sum_{u\in\mathcal{U}} \bar{b}^{(u)} - \bar{y}_g P)]^+ \tag{7.5}$$

with $\alpha > 0$, $\lambda_1 = 0$ and $0 \leq \beta < 1$. Update (7.3)-(7.4) is a Frank-Wolfe descent update while (7.5) is a dual subgradient ascent update and the joint convergence of these to a ball around the optimum is established by, for example, Corollary 2 in [71]. Note that in 7.5 we use the running average for the dual update, but as it has been proved in [71], the iteration still converges the optimum.

However, since the Lagrangian is linear in x_g the solution x_g in (7.3) is either 0 or y_{max} *i.e.* we end up with a bang-bang type of solution whose running average $\bar{y}_g$ converges to the optimum. Such bang-bang scheduling of packets is not well suited to Internet applications since it is bursty and can lead to large fluctuations in delay. We therefore consider the following modification to the update:

$$\delta\bar{y}_g \in \arg\min_{x\in[-1,1]}(\gamma - \lambda_g P)x \tag{7.6}$$

$$\bar{y}_g = [\bar{y}_{g-1} + \beta\delta\bar{y}_g]^{[0,y_{max}]} \tag{7.7}$$

$$\lambda_{g+1} = [\lambda_g + \alpha(\sum_{u\in\mathcal{U}} \bar{b}^{(u)} - \bar{y}_g P)]^+ \tag{7.8}$$

where $[y]^{[0,y_{max}]} = y$ when $y \in [0, y_{max}]$, 0 when $y < 0$ and y_{max} when $y > y_{max}$.

Observe that $|\lambda_{g+1} - \lambda_g| \leq \alpha|\sum_{u\in\mathcal{U}} \bar{b}^{(u)} - \bar{y}_g P| \leq \alpha b_{max}$ where $b_{max} := \sum_{u\in\mathcal{U}} \bar{b}^{(u)}$. The following lemma now establishes that update (7.6)-(7.7) ensures that λ_g is bounded:

Lemma 2 (Boundedness)**.** Consider update (7.6)-(7.7). Suppose $|\lambda_{g+1}-\lambda_g| \leq \epsilon/(2Py_{max})$, $\epsilon > 0$, and $y_{max} > b_{max}$ is sufficiently large. Then as $g \to \infty$ we have $|\gamma - \lambda_g P| \leq \epsilon' := (2\beta + 1)\epsilon/\beta$.

Proof. See Appendix A. □

Selecting step size $\alpha \leq \epsilon/(2Py_{max}^2)$ is sufficient to satisfy the conditions of this lemma. Now,

$$\frac{1}{g}\sum_{i=1}^{g-1}(\sum_{u\in\mathcal{U}} \bar{b}^{(u)} - \bar{y}_g P) = \sum_{u\in\mathcal{U}} \bar{b}^{(u)} - y^\diamond P \leq \frac{\lambda_g}{\alpha g} \tag{7.9}$$

and since λ_g is bounded then $\sum_{u\in\mathcal{U}} \bar{b}^{(u)} \leq y_g^\diamond P$ as $g \to \infty$. That is, the mean service rate provided by the traces is sufficient to serve the mean rate of user packet arrivals.

Update (7.6)-(7.8) requires knowledge of the long-term average arrival rate $\bar{b}^{(u)}$ to carry out step (7.8), which we do not have. Also, the average number of active traces $\bar{y}_g$ is real valued whereas the number of active traces, which is the quantity needed in order to implement a scheduler, is integer valued. To address these issues we therefore consider the following update:

$$\delta y_g \in \arg\min_{x\in[-1,1]}(\gamma - \tilde{\lambda}_g P)x \tag{7.10}$$

$$y_{g+1} = [y_g + \delta y_g]^{[0,y_{max}]} \tag{7.11}$$

$$\tilde{\lambda}_{g+1} = [\tilde{\lambda}_g + \alpha(\sum_{u\in\mathcal{U}} b_g^{(u)} - y_g P)]^+ \tag{7.12}$$

Observe that the packet arrivals at time g are now used in (7.12) and so knowledge of the mean arrival rate $\bar{b}^{(u)}$ is *not* required. Also, the number of active flows y_g is now used rather than the average $\bar{y}_g$ (although observe that this just amounts to selecting $\beta = 1$). It follows immediately from Lemma 2 that $\tilde{\lambda}_g$ is bounded provided the packet arrivals $\sum_{u\in\mathcal{U}} b_g^{(u)}$ in an interval are bounded, y_{max} is sufficiently large and step size α is selected sufficiently small.

Discussion

The algorithm (7.10)-(7.12) is intuitive. Multiplier $\tilde{\lambda}_g$ measures the accumulated mismatch between the arrival of user packets and service of packets ($\sum_{u\in\mathcal{U}} b_g^{(u)}$ is the number of packet arrivals in interval g, y_g is the number of traces active in interval g and $y_g P$ is therefore the number of packets served over the interval). When this mismatch grows too large and exceeds γ/P then we increase the number of active traces by one, when the mismatch falls to than γ/P we decrease the number of active traces. That is, the scheduler therefore uses a threshold rule.

As we will shortly see, the value of $\tilde{\lambda}_g$ is closely related to the occupancy of the queues at the input to the scheduler (at the left-hand side of Fig 7.5) and so in effect the scheduler operates by activating new traces when the queue backlog becomes too large. This also suggests that there is a trade-off between delay and optimality (*i.e.* throughput efficiency). That is, a large queue backlog means that there are plenty of user packets available to be scheduled and hardly any dummy packets are needed, whereas a smaller queue means the scheduler is more likely to need to insert dummy packets in order to ensure transmission obey the specified trace profile. However, a large queue implies large delay.

Queue Stability

Update (7.10)-(7.12) stabilises the input queue to the scheduler. The occupancy of the input queue to the scheduler satisfies the following

$$q_{gn+1} = [q_{gn} + \sum_{u\in\mathcal{U}} b_g^{(u)} - y_g P]^+ \tag{7.13}$$

at the time slots $k = gn$, $g = 1, 2, \ldots$ corresponding to the end of each group g of slots (so we are looking the queue occupancy in embedded time). Observe that letting

$\tilde{q}_g := \tilde{\lambda}_g/\alpha$ then by update (7.12)

$$\tilde{q}_{g+1} = [\tilde{q}_g + \sum_{u\in\mathcal{U}} b_g^{(u)} - y_g P]^{[0,y_{max}]} \tag{7.14}$$

which, apart from time dilation by n and the upper limit y_{max}, is identical to (7.14). Hence, when y_{max} is sufficiently large we can identify $\tilde{\lambda}_g/\alpha$ with q_{gn}. By Lemma 2 we therefore have that

$$q_g \leq \frac{\gamma + \epsilon'}{P\alpha} \tag{7.15}$$

for g sufficiently large and so the input queue is uniformly bounded (so stable). Further, it can be seen that the bound on the queue occupancy is proportional to γ and $1/\alpha$ (so decreasing γ or increasing α decreases the queue occupancy) but inversely proportional to the trace length P (so longer traces reduce the queue occupancy).

7.3.3 Unsynchronised Scheduler

We now relax the synchronisation assumption made in the previous section and allow new traces to start in any slot. This means that traces can, for example, now partially overlap. The payoff is that the potential exists to exploit this extra freedom to achieve improved performance, especially improved delay performance as a result of being able to start new traces in a more timely way.

As our baseline unsynchronised scheduler we use the following update,

$$\delta x_k \in \arg\min_{x\in[-1,1]} (\gamma - \hat{\lambda}_k P)x \tag{7.16}$$

$$x_{k+1} = [x_k + \delta x_k]^{[0,y_{max}]} \tag{7.17}$$

$$\hat{\lambda}_{k+1} = [\hat{\lambda}_k + \alpha(\sum_{u\in\mathcal{U}} a_k^{(u)} - \sum_{t\in\mathcal{T}_k} p_{k-\tau_t})]^+ \tag{7.18}$$

where a new trace is started in slot k (added to $\mathcal{T}_k$) when x_k increases or when a trace completes and the number of active traces $|\mathcal{T}_k|$ falls below x_k.

Observe that $|\lambda_{g+1} - \lambda_g| \leq \alpha|\sum_{u\in\mathcal{U}} a_k^{(u)} - \sum_{t\in\mathcal{T}_k} p_{k-\tau_t}|$. Assuming that the number of packets arrivals $\sum_{u\in\mathcal{U}} a_k^{(u)}$ is bounded it follows from Lemma 2 that $\hat{\lambda}_k$ is bounded provided y_{max} is sufficiently large and α is sufficiently small.

Similarly to the synchronised case,

$$\frac{1}{k}\sum_{i=1}^{k}(\sum_{u\in\mathcal{U}} a_i^{(u)} - \sum_{t\in\mathcal{T}_i} p_{i-\tau_t}) \leq \frac{\tilde{\lambda}_k}{\alpha k} \tag{7.19}$$

Now, $\frac{1}{k}\sum_{i=1}^{k}\sum_{u\in\mathcal{U}} a_i^{(u)} \to \sum_{u\in\mathcal{U}} \bar{b}^{(u)}$ as $g \to \infty$. Since $\tilde{\lambda}_k$ is bounded then $\sum_{u\in\mathcal{U}} \bar{b}^{(u)} \leq \frac{1}{k}\sum_{i=1}^{k}\sum_{t\in\mathcal{T}_i} p_{i-\tau_t}$ as $g \to \infty$. That is, the mean service rate provided by the traces is sufficient to serve the mean rate of user packet arrivals.

7.4 Experimental Results

We built a prototype VPN implementation of the scheduler in (7.10)-(7.12) and also of its unsynchronised variant. Using this VPN prototype, in this section we present experimental measurements of the scheduler performance under a wide range of conditions. In particular, while the analysis in the previous section focusses on throughput efficiency, our measurements allow us to also measure the delay performance of the scheduler and the impact of design choices (such as the use of unsynchronised trace activation) on this. We present results for both UDP and TCP traffic, a potential concern with TCP being possible interactions between the action of its congestion control and of the VPN scheduler (since both affect packet rate). In addition we present measurements for web page fetches carried out over the VPN and evaluate the delay and throughput efficiency achieved.

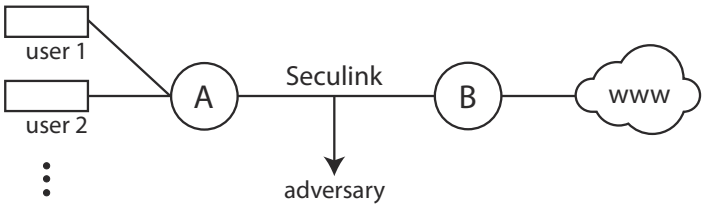


Fig. 7.7 Schematic illustrating the tunnel setup considered.

7.4.1 Hardware Setup

We begin by describing the hardware setup used. The network topology is shown schematically in Figure 7.7. Clients are connected to the VPN gateway A using an IPSec protected link. The link between nodes A and B (labelled Seculink) is the encrypted privacy-enhanced VPN tunnel. This is a public link, *i.e.* it is exposed to sniffing from adversaries. Seculink gateway A is a commodity server with an Intel(R)

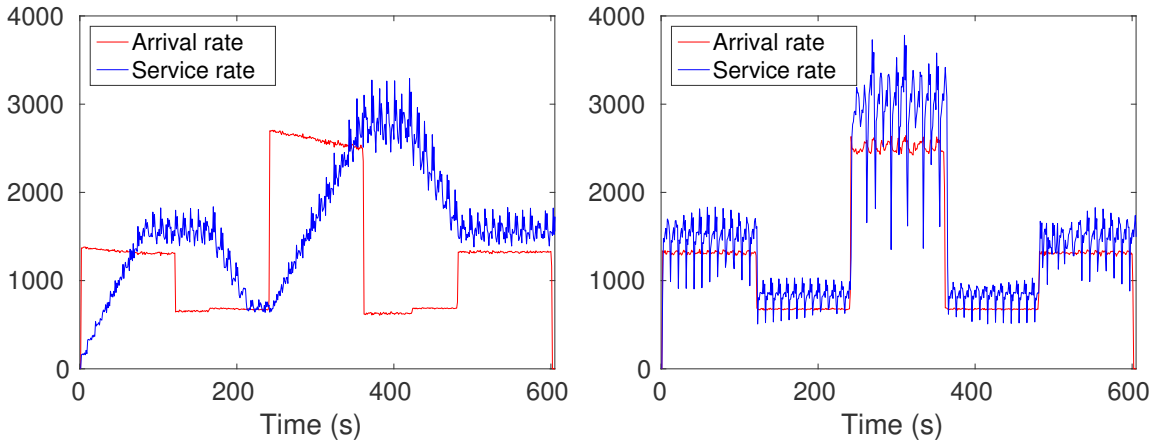
Core 2 Quad @2.66GHz CPU and 4GB RAM running Ubuntu 14.04.4 LTS. The other end of the tunnel, node B, is another commodity server with an Intel(R) Core 2 Duo @ 3.00GHz and 2GB RAM running Ubuntu 12.04.5 LTS. Nodes A and B use a mix of TP-LINK TG-3468 1000Mbps external ethernet cards and on-board gigabit ethernet cards (RTL8111 PCI Express and Intel 82567LM-3 respectively) for network connectivity. Traffic from node B is routed to the Internet via a campus gateway using a 100Mbps link (it is this link which limits the network rate since the tunnel between A and B is a gigabit connection). A NETGEAR JGS516 Gigabit switch is used to connect client machines to Seculink gateway A. The Seculink tunnel uses IPSec and traffic shaping to protect against DPI and traffic analysis attacks. Outgoing traffic from clients is sent to node B via node A and then forwarded by B to the campus gateway. Incoming traffic arriving at node B is forwarded to node A and then sent to the corresponding client.

7.4.2 Software Setup

Both Seculink and links between clients and node A are protected with IPSec protocols AES-256 and SHA-256. Traffic shaping on Seculink is implemented using the `netfilter` and `netfilter_queue` libraries in C in conjunction with the `iptables` module in Linux. Gateway A queues packets from clients and transmits them over Seculink using a traffic shaping scheduler. When no client packets are queued for transmission yet the active traces require a packet to be sent then a dummy packet is generated and sent. Traffic received at node B is first filtered to remove dummy packets and remaining packets are then forwarded to the campus gateway over an unencrypted link. Responses received from Internet are treated similarly, transmitted by node B to node A using the traffic shaping scheduler. In the UDP experiments traffic is generated using the PERIODIC method of the MGEN traffic generator. TCP traffic is generated by using `wget` to fetch dummy files of various sizes from a server located in the campus network.

7.4.3 Synchronised vs Unsynchronised Schedulers

With a synchronised scheduler traces only start and finish at the beginning of each cycle, a cycle being $9s$ duration in all of our tests. This means that when the arrival rate is changed in the middle of a cycle we have to wait until the start of a cycle to update the number of active traces. Hence, when new flows start they may experience significant delay. This is particularly an issue when no traces are active in the tunnel



(a) Synchronised (b) Unsynchronised

Fig. 7.8 Comparison of tunnel rate adaptation in synchronised and unsynchronised scenarios.α in synchronised scenario. (duration: 10min, $\alpha = 1$, $P = 1682, n = 9615, c = 20, \gamma = 100$)

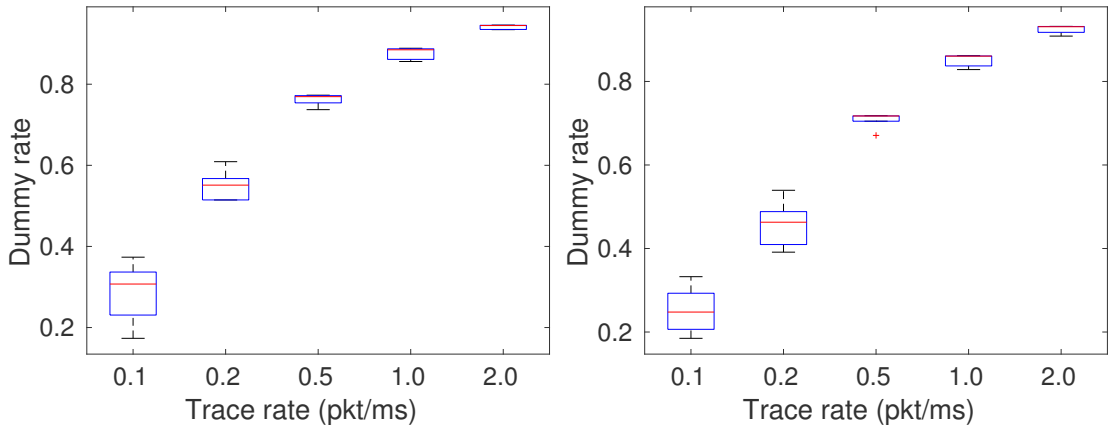
and a new client flow starts since in this case the scheduler cannot transmit any packets until the next cycle starts (when some traces are already active the scheduler has the option to transmit some packets for the new flow *e.g.* by substituting for packets from other flows or for dummy packets). Similarly, when the rate of incoming traffic is less than the rate of a single trace then sufficient packets need to be queued before a trace is activated, causing delay.

To mitigate such delays on lightly loaded links, two unsynchronised modifications are implemented to "wake" the channel from silence upon sensing new incoming traffic:

1. *DNS Triggering.* Upon receiving a DNS packet when no traces are active, a new trace is immediately activated.

2. *Use of Running Average When Lightly-Loaded.* We maintain a running average of packet arrivals, $\bar{a} = (1-\zeta)\bar{a} + \zeta \sum_{u \in \mathcal{U}} a_k^{(u)}$, where $\zeta > 0$ is a design parameter. When $\bar{a}$ exceeds threshold a^* and no traces are active then a new trace is activated. In our experiments we use $\zeta = 0.001$, $a^* = 0.005$.

In addition, to avoid adding new traces in response to small spikes in arrivals, when $\gamma - \hat{\lambda}_k P < 0$ we only activate a new trace when also $\hat{\lambda}_k - \hat{\lambda}_{k-m} > 0$, where m is a parameter. In this way we only respond to a sustained increase in $\hat{\lambda}$. Similarly, when $\gamma - \hat{\lambda}_k P > 0$ we only decrease x_k when also $\hat{\lambda}_k - \hat{\lambda}_{k-m} < 0$. In our experiments we use $m = 100$ unless otherwise stated, which is found to provide a good compromise between responsiveness and sensitivity to small fluctuations.



(a) Uplink (b) Downlink

Fig. 7.9 Dummy overhead vs trace rate for a sample website. For each rate data is shown for traces of duration 1, 2, 4, 8 and 16 seconds. ($\gamma = 4096$).

To reduce sensitivity to small fluctuations in arrivals we also modify the $\tilde{\lambda}$ update to use $\max\{\bar{a}, \sum_{u\in\mathcal{U}} a_k^{(u)}\}$, namely:

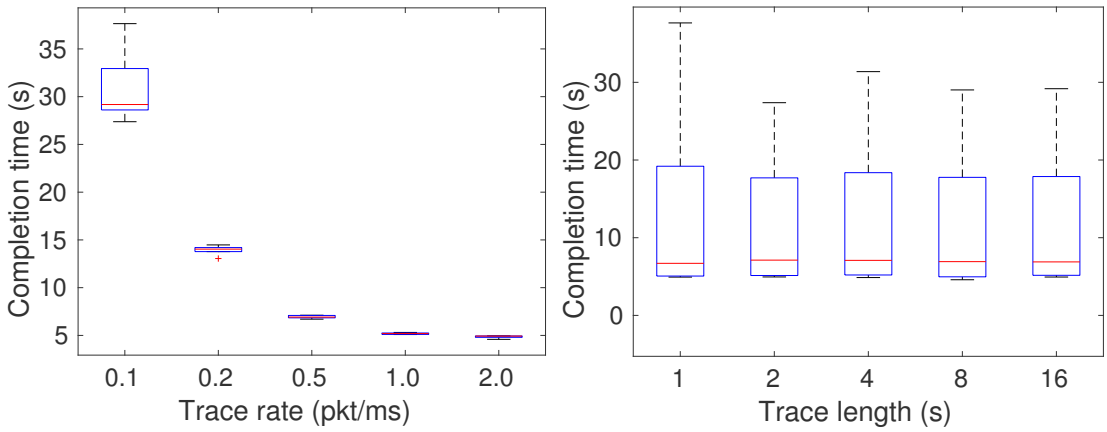
$$\hat{\lambda}_{k+1} = [\hat{\lambda}_k + \alpha(\max\{\bar{a}, \sum_{u\in\mathcal{U}} a_k^{(u)}\} - \sum_{t\in\mathcal{T}_k} p_{k-\tau_t})]^+ \tag{7.20}$$

In this way when the number of packet arrivals falls temporarily $\bar{a}$ is used instead of a_k.

Figure 7.8 illustrates the impact of these changes, comparing time histories of the arrival and services rates when using a synchronised scheduler vs an unsynchronised scheduler. It can be seen that the unsynchronised scheduler is much more responsive to changes to traffic load. Not only is the delay in increasing the service rate in response to increases in arrival rate reduced (so reducing the delay experienced by user packets) but also the delay in reducing the service rate in response to a fall in the arrival rate is also reduced (so reducing the number of dummy packets transmitted). Since it has better performance, in the rest of this section we will make use of this unsynchronised scheduler unless otherwise stated.

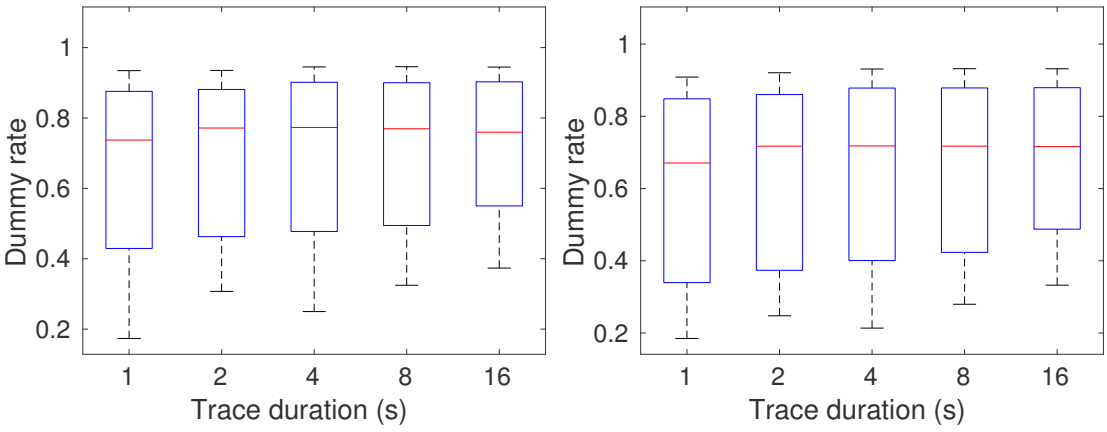
7.4.4 Choice of Trace Parameters

We make use of traces which transmit at a constant rate for a defined duration. Figure 7.9 shows measurements of the delay and fraction of dummy packets vs the trace rate and duration. This data is for an example web fetch using TCP and shows



(a) Different rates (b) Different durations

Fig. 7.10 Illustrating the effect of trace rate and duration on the completion time of a sample web page. For each rate the data shown in (a) is for traces of duration 1, 2, 4, 8 and 16 second. Similarly, for each duration the data shown in (b) is for rates of 0.1, 0.2, 0.5, 1.0 and 2.0 pkts/ms. ($\gamma = 4096$).



(a) Uplink (b) Downlink

Fig. 7.11 Illustrating the effect of trace duration on dummy overhead for a sample website. For each duration the data shown for rates of 0.1, 0.2, 0.5, 1.0 and 2.0 pkts/ms. ($\gamma = 4096$).

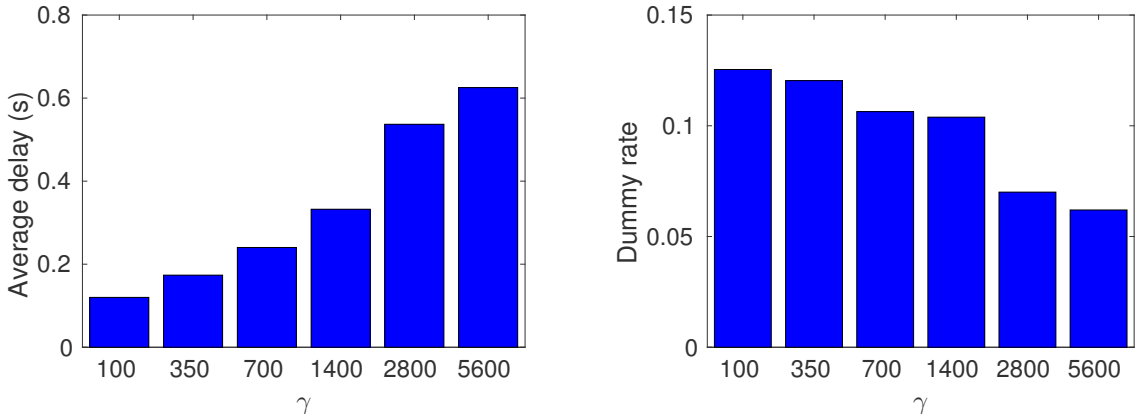


Fig. 7.12 Comparison between dummy and delay rates for CBR UDP traffic and different choices of γ. Given the negligible variance, average over 5 tests are presented for each γ. (duration: 5min, $\bar{a} = 2800$, $\alpha = 1$, $P = 1682, n = 9615, c = 20$). Values for each γ are averaged over 5 tests (error bars are not shown since they are too small to be clearly visible).

measurements for both the downlink and uplink (the uplink carries the TCP ACKs). Figure 7.10 shows the corresponding impact of the trace rate and duration on the web fetch completion time. It can be seen from these plots that increasing the trace rate reduces completion time but increases the dummy packet overhead, but the duration of trace has limited effect on completion time. The dummy packet overhead for different trace durations is also shown in Figure 7.11. From now on we use a trace of duration 9s and rate 0.17 pkts/ms, with aim of achieving a reasonable balance between delay and dummy packet overhead.

7.4.5 Performance with CBR UDP Traffic

In this section we study throughput efficiency and delay performance with constant rate UDP arrivals. By throughput efficiency we mean the mismatch between the transmit rate of the scheduler and the arrival rate of client packets. Recall that when there are insufficient client packets buffered at the scheduler then the scheduler transmits dummy packets so that its transmissions can continue to follow the predefined trace pattern. These dummy packets provide resistance to traffic analysis attacks but increase the load on the network and so we would like to minimise these. There is a fairly direct trade-off between delay and the volume of dummy packets transmitted since by increasing the backlog of client packets buffered at the input to the scheduler we reduce the need for dummy packets but increase the delay experience by client

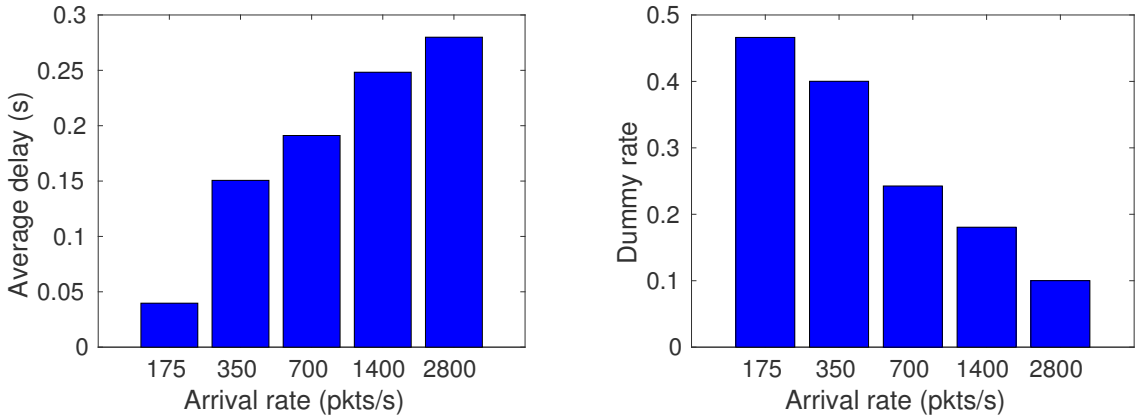
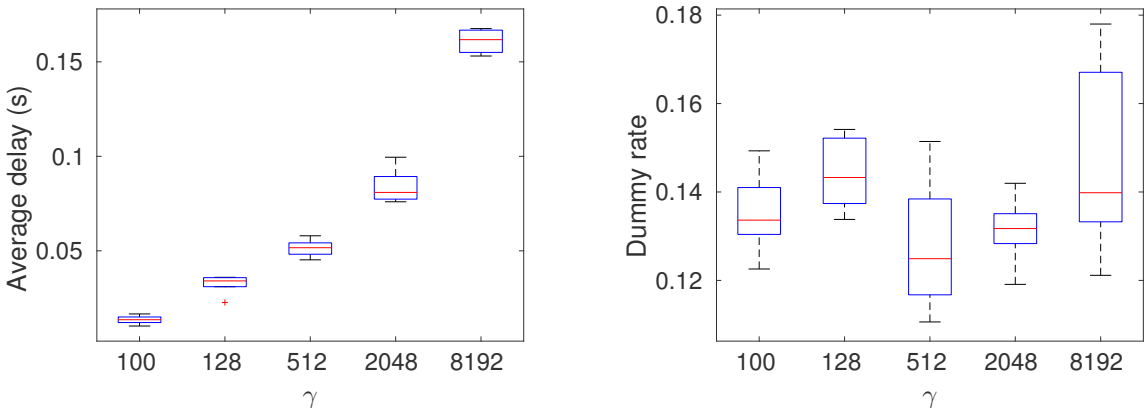


Fig. 7.13 Comparison between delay and dummy rates vs arrival rate for CBR UDP traffic with $\gamma = 1024$. (duration: 5min, $\alpha = 1$, $P = 1682, n = 9615, c = 20$). Values are averaged over 5 tests.

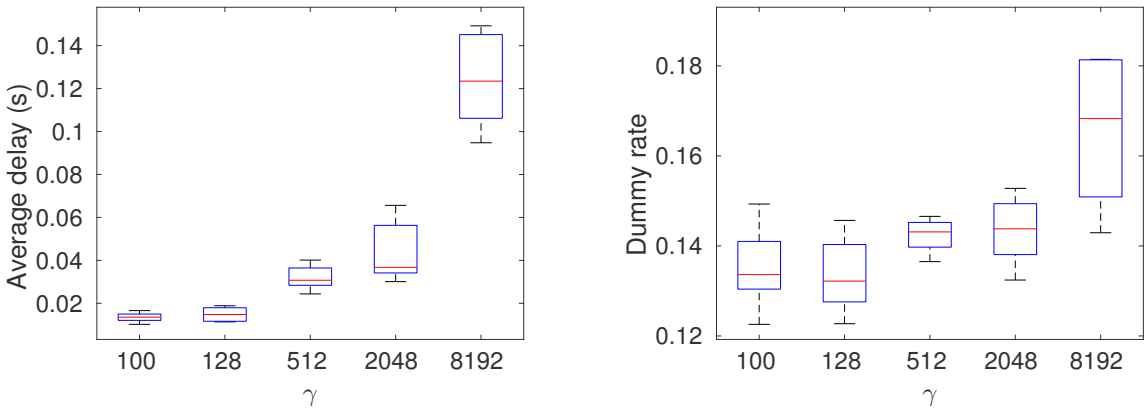
packets. Conversely, reducing the number of client packets buffered tends to increase the need for dummy packets but decrease delay.

This trade-off between throughput efficiency and delay can be seen in Figure 7.12, which plots measurements of the delay and dummy rate vs scheduler parameter γ which directly influences the queue backlog (with increases in γ increasing the backlog). The dummy rate value shown is the ratio of the dummy packets sent to the total number of packets sent (dummy plus processed packets). Note that there is no packet loss within the scheduler in these tests. It can be seen from Figure 7.12 that as γ is increased the delay rises but the rate at which dummy packets are sent falls. Observe, however, that even for relatively small values of γ the dummy rate remains reasonably small, which is encouraging.

Figure 7.13 shows corresponding measurements of delay and dummy rate as the arrival rate is varied (scheduler parameter γ is held constant at 1024 in these plots). It can be seen that the delay tends to increase with arrival rate and the dummy rate to fall. This can be understood by noting that as the arrival rate rises more user packets tend to be buffered at the scheduler. Hence, the delay rises but also user packets are more likely to be available when a transmission opportunity occurs and so the number of dummy packets needed falls. Importantly, observe that the delay is consistently reasonable, rising to no more than 275ms at higher arrival rates, despite the extensive traffic shaping being carried out. Also, the decrease in dummy rate with increasing arrival rate means that the throughput efficiency increases with increasing load, so mitigating the dummy packet cost incurred by the traffic shaping.



(a) Uplink



(b) Downlink

Fig. 7.14 Measurements of delay and fraction of dummy packets vs γ for TCP traffic. The data is measured fetching a 1024MB file, the average over 5 tests is presented for each γ ($\alpha = 1$, $P = 1682, n = 9615, c = 20$).

7.4.6 Performance with TCP Flows

We now present data on the scheduler performance with TCP traffic (for the default Linux Cubic TCP variant). Figure 7.14 plots measurements of delay and fraction of dummy packets vs design parameter γ when fetching a 1024MB file from a campus server using `wget`. Data is shown for both the downlink and uplink, the packet rate on the uplink being roughly half of that on the downlink due to delayed acking by TCP. It can be seen that the delay increases with γ, as expected, but remains less than 100ms even for relatively large values of γ. The fraction of dummy packets is relatively insensitive to γ when using TCP.

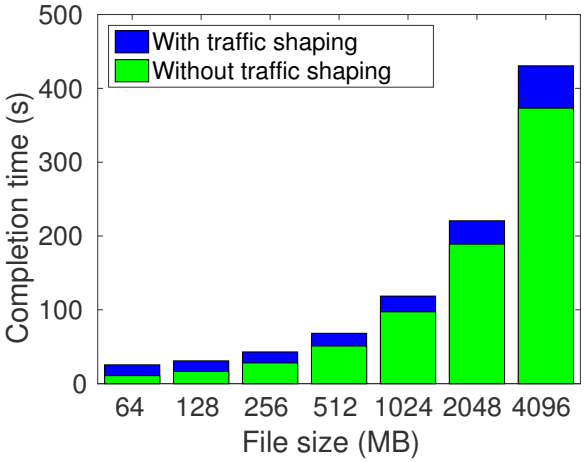
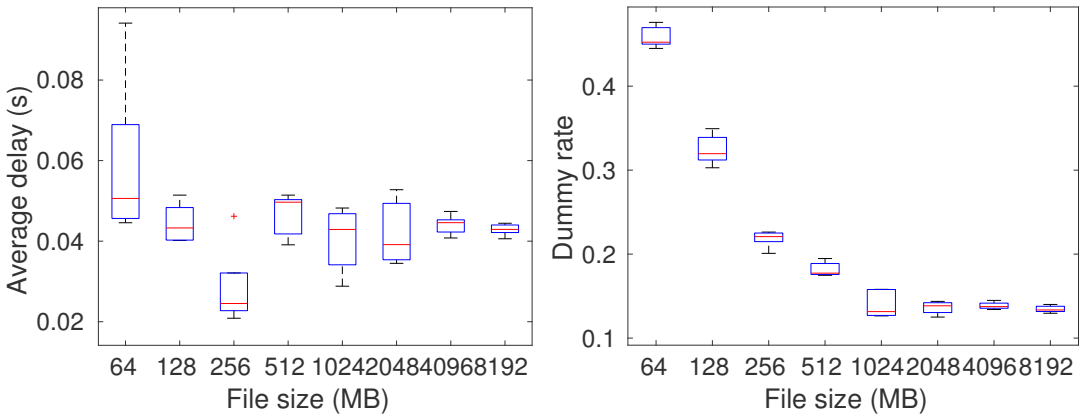


Fig. 7.15 Completion time vs file sizes when using TCP, $\gamma = 1024$. Results are shown both for the traffic shaped tunnel and for an unprotected link (no encryption, no traffic shaping). Values are averaged over 5 runs.

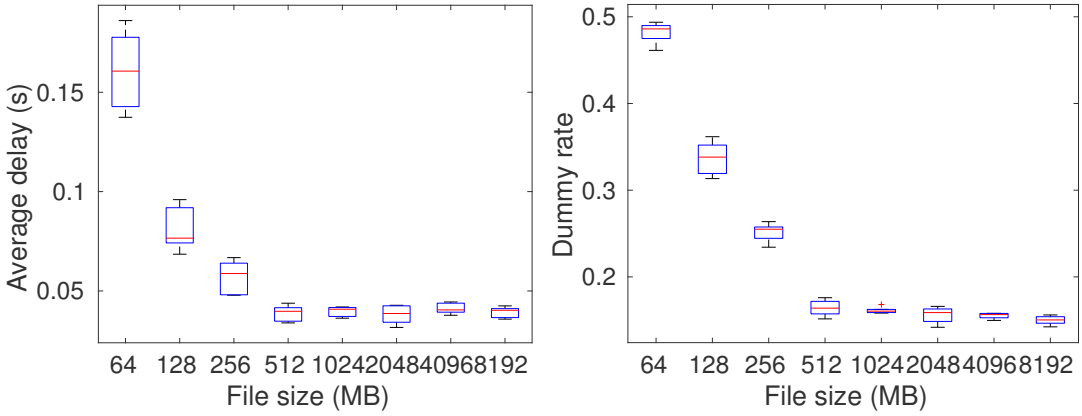
Figure 7.15 plots the measured completion time vs file size fetched. For comparison, the corresponding data is also shown for a link with no traffic shaping. It can be seen that the cost, in terms of increase in completion time, is modest. Figure 7.16 provides additional detail, showing measured data on delay and dummy rate vs file size. It can be seen that the delay and dummy rate both tend to fall as the file size increases. The effect here is due to the time that it takes the scheduler to adapt to the arrival of a new flow: for longer flows this adaptation overhead gets washed out and amortised over many packets but for short flows its effect is more pronounced. This can be seen in Figure 7.17, which shows time histories of the number of active traces on both the downlink and uplink (uplink traces are carrying the TCP acks and so are fewer in number). The experiment is conducted by fetching a file of size 8192MB. The link reaches steady state in about 20 seconds.

7.4.7 Web Traffic Privacy

We fetched the home pages from the alexa top 100 finance and health web sites in Ireland. Figure 7.18 shows four example trace time histories recorded during these fetches. It can be seen that the traces time histories in Figures 7.18a and 7.18c are identical and so evidently these two web pages cannot be distinguished by an attacker. Figure 7.19 plots the number of distinct trace time histories measured on the uplink and downlink while fetching the 100 web pages and also the number of pages for which each trace time history is observed. It can be seen that certain trace time histories are



(a) Uplink



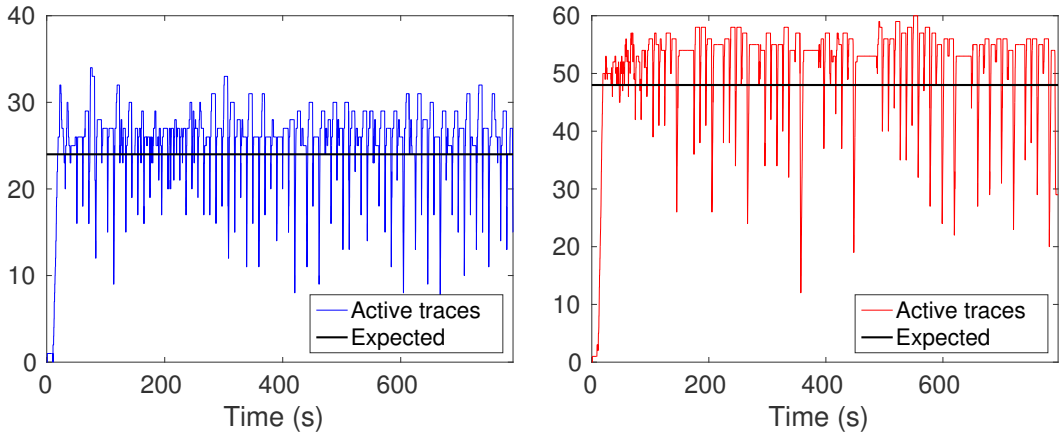
(b) Downlink

Fig. 7.16 Comparison between dummy and delay rate when using TCP to fetch files of different sizes, for $\gamma = 1024$. Box plots for values of 5 runs are shown.

generated by 10-30 different web pages and so these web pages are indistinguishable to an attacker.

It can also be seen in Figure 7.19 that around 20 trace time histories are generated by a single web page, and so potentially vulnerable to attack. An example is shown in Figure 7.18d. Evidently the trace time history in Figure 7.18d cannot be distinguished from *two* fetches of the web sites in Figures 7.18a and 7.18c. The trace time history in Figure 7.18b is more complex, and raises the question of whether there exists one or more combinations of web page fetches that yield the same trace time history and so cannot be distinguished from this web fetch.

We proceed as follows. Let $\mathcal{H}$ denote the set of possible trace time histories. For trace time history $h \in \mathcal{H}$ we determine (by exhaustive search) the combinations



(a) Uplink (b) Downlink

Fig. 7.17 Illustrating convergence speed of active traces for uplink and downlink. The active traces approach channel capacity only 20 seconds into the fetch. Note that expected value of uplink capacity is half the value of downlink and that is due to TCP protocol sending one ACK for every two data packets.

$C_{h,1}, C_{h,2}, \ldots, C_{h,m_h}$ of the other trace time histories that are indistinguishable from h. Combination $C_{h,i} = \{(h_{i,1}, n_1), (h_{i,2}, n_2), \ldots\}$ with $h_{i,j} \in \mathcal{H} \setminus \{h\}$ and n_j the number of times that $h_{i,j}$ is repeated in the combination. Let $\mathcal{W}_h$ denote the set of web pages that generate trace time history h, so $|\mathcal{W}_h|$ is the number of web pages that generate h (see Figure 7.19). Let the web page fetched W be a random variable that takes values in $\cup_{h\in\mathcal{H}}\mathcal{W}_h$. Assume, for simplicity, that the web pages in W_h are equally likely to be fetched. Then when combination $C_{h,i}$ is observed the probability that page ω was fetched is,

$$\begin{aligned} &\mathrm{P}(W = \omega | C_{h,i}) \\ &\quad = \sum_{(h,n)\in C_i : \omega \in h} \sum_{j=1}^{n} \binom{n}{j} \frac{1}{|\mathcal{W}_h|}^{j} (1 - \frac{1}{|\mathcal{W}_h|})^{n-j} \end{aligned} \tag{7.21}$$

Let C be a random variable which is the combination used. Assume, again for simplicity, that each combination $C_{h,1}, C_{h,2}, \ldots, C_{h,m_h}$ is equally probable when trace h is observed, i.e $\mathrm{P}(C = C_{h,i}) = 1/m_h$. Then the probability that web page ω was fetched given that trace time history h was observed is,

$$\mathrm{P}(W = \omega | h) = \sum_{i=1}^{m_h} \mathrm{P}(W = \omega | C_{h,i})/m_h \tag{7.22}$$

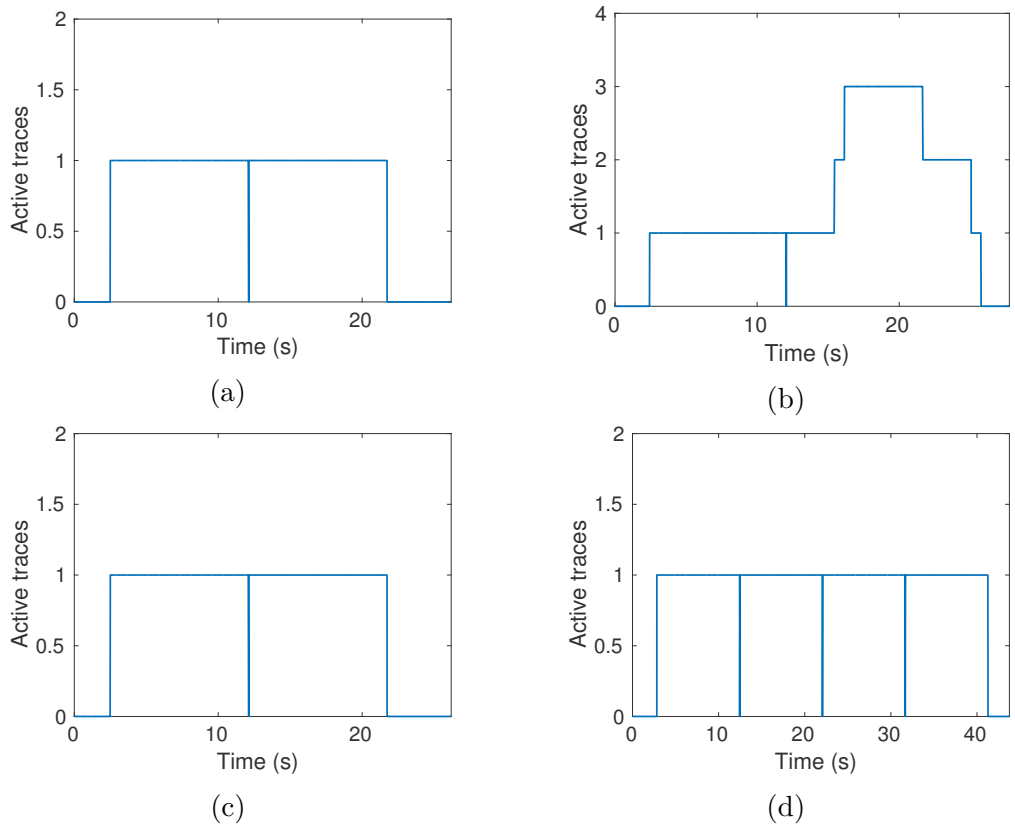
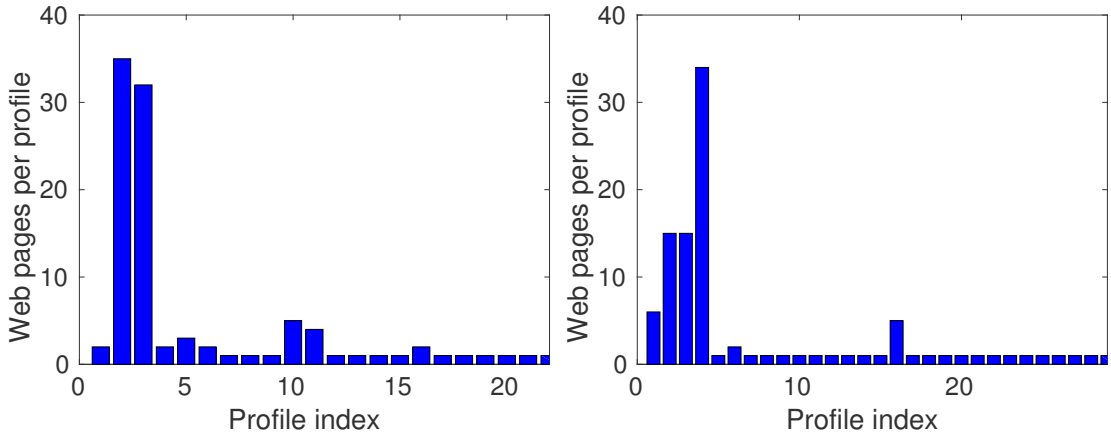


Fig. 7.18 Examples of trace time histories recorded when fetching the home pages of four websites from the alexa top 100 finance and health web sites in Ireland. ($P = 1682, n = 9615, c = 20, \gamma = 1024$)

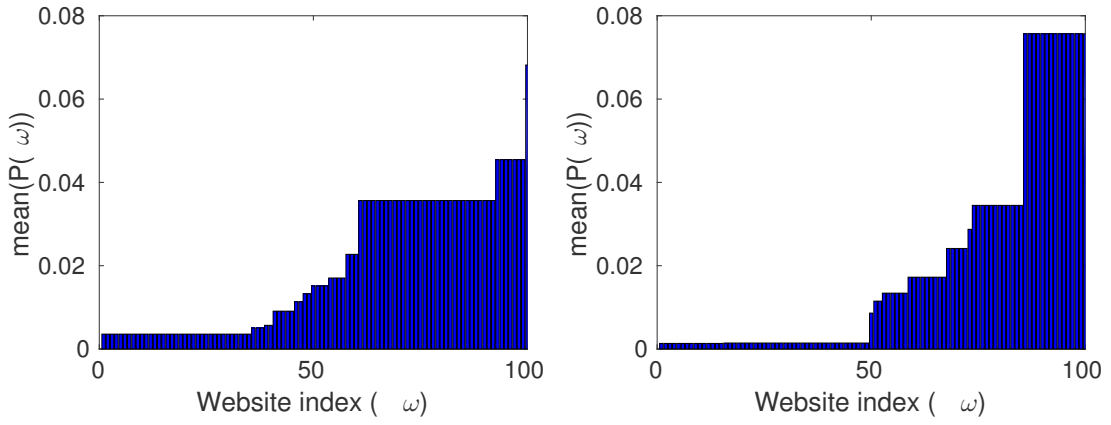
When the traces $h \in \mathcal{H}$ are equally likely to be observed then $\mathrm{P}(W = \omega) = \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \mathrm{P}(W = \omega | h)$. Figure 7.20 plots $\mathrm{P}(W = \omega)$ calculated using (7.22) for each of the 100 web pages, sorted in increasing order. It can be seen that this probability is less than 0.08 on the downlink and less than 0.05 on the uplink. We can conclude therefore that with the foregoing assumptions the user can reasonably deny that page ω was fetched despite an attacker observing the transmitted packet trace.

To get a sense of the sensitivity of these values to the assumption of equiprobability, for comparison, let $h^* \in \arg\min_{h \in \mathcal{H}} \mathrm{P}(W = \omega | h)$ and suppose that this worst case trace h^* is always observed. Then $\mathrm{P}(W = \omega) = \mathrm{P}(W = \omega | h^*)$. Figure 7.21 shows the calculated $\mathrm{P}(\omega | h^*)$ for our measured web fetches. It can be seen that $\mathrm{P}(\omega | h^*)$ is higher than in Figure 7.20, as expected, and indeed reaches one but only for a small number of web pages. Hence, even in this worst case a user has fairly level of strong deniability.



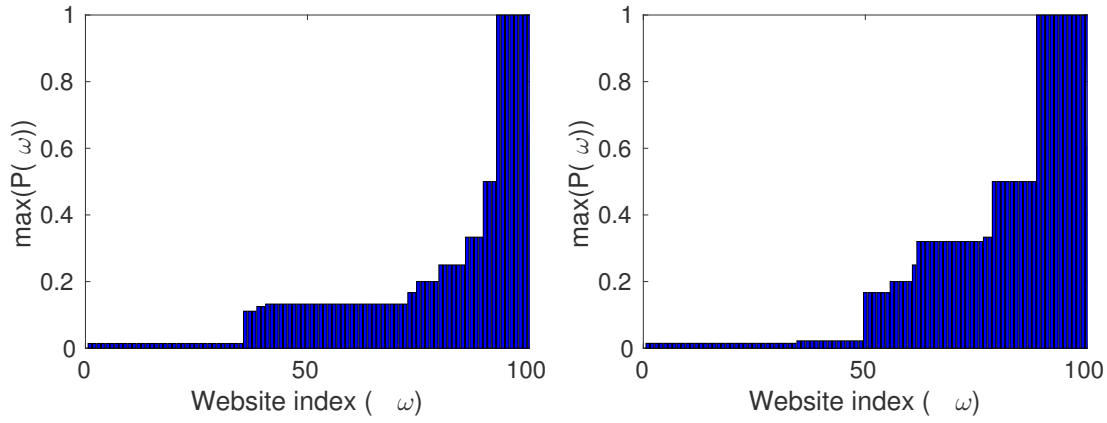
(a) Uplink (b) Downlink

Fig. 7.19 Counts of the number of web pages generating each distinct web trace time history.



(a) Uplink (b) Downlink

Fig. 7.20 Probability $P(\omega)$ that web page ω was fetched given an observed trace time history vs ω and assuming equiprobable combinations.



(a) Uplink (b) Downlink

Fig. 7.21 Probability P(ω) that web page ω was fetched given an observed trace time history vs ω and assuming worst combination has probability one.

7.5 Summary and Conclusions

In this chapter we introduce a trace-based tunnel that is resistant to traffic analysis in the sense that it provides deniability to users that any specified web page was fetched given that a specified packet trace is observed on the tunnel. We present a scheduler design for managing the transmission of traces to satisfy user traffic demand while maintaining reasonably low delay and throughput overhead due to dummy packets. Experimental results are also presented demonstrating the effectiveness of this scheduler under a range of realistic network conditions and real web page fetches.

Chapter 8

Conclusion

8.1 Summary

In this thesis, we study traffic analysis attacks and defences against web traffic. We begin by introducing an attack against web traffic that enables an adversary to guess with a high probability the websites being fetched by a user over the internet. The attack uses only the inter-arrival times between packets and thus is immune against the state of the art defences in the literature.

We present a fair rate allocation defence against Bernoulli arrivals. This is then extended to a more practical defence model that considers general arrival processes. The defence serves the incoming traffic by transmitting traces of fixed rate and length to obfuscate the timing patterns. The number of active traces is optimized to correspond to arrival rates reducing bandwidth overhead. Being a multi-user tunnel, it may also transmit packets from other users instead of dummy traffic to improve the rate of unnecessary dummy traffic. The trade-off between privacy of defence and the overhead on bandwidth and delay is also thoroughly evaluated.

8.2 Discussion

The attacks and defences proposed in this thesis overcome the limitations of state of the art privacy-enhancing technologies. To address these limitations, we considered the worst case scenario where all existing mitigating technologies are already applied to the network link. This includes encryption, packet padding, and dummy traffic insertion so that the start and end of web traces are obfuscated. We also assume that the adversary has the ability to extract individual user traffic traces from a mixed

traffic in a shared network. This presents an opportunity to further improve the attacks and defense mechanisms introduced, in the absence of these restrictions:

- The attack investigated here is assumed to have limited information about incoming traffic. However knowing more about traffic patterns may prove useful for improving performance in classifying web pages: for example, knowing the beginning and end of each web page fetch, which as discussed in Chapter 4 provides better success rates in identifying web pages. Inference methods are also needed when multiple web pages are fetched simultaneously, with mixture models being the natural starting point for this.

- The defences also benefit from additional information from upper network layers. Particularly by knowing the type of protocol used by the client, a number of modifications such as early trace activation (as explained in Chapter 7 or using update 7.20, that were specifically proposed to overcome TCP congestion control can be disabled for other types of traffic *e.g.* UDP, improving dummy overhead and delay. Further, if the type of service/application in used is known, a different choice of γ for services like file download or video streaming leads to lower dummy overhead.

- The use of traces with periodic rate and fixed duration was introduced to provide a clear understanding of service rate at every time slot. However, employing a more complicated trace like that of an actual web fetch might have a better dummy and delay performance. This of course requires a more intelligent method of trace activation to synchronize the beginning of traces with fetch requests in both uplink and downlink nodes.

8.3 Future Work

The purpose of this thesis is to address the deficiencies of current privacy-enhancing technologies and introduce a defence that mitigates traffic analysis attacks with little impact on quality of experience. The prototype presented provides guidelines for creating a privacy-enhanced VPN that allows users to control how much privacy they are willing to sacrifice in exchange for quality of experience. While we investigated a balance state between privacy, delay and throughput overhead, our approach addresses a trade-off between these overheads and privacy which allows users to opt in for lower delay but higher dummy and lower privacy, or high privacy with cost of additional delay by modifying γ.

The attacks and counter measures considered in this thesis are at Internet layer i.e. the mitigation and traffic morphing happens at packet level. This allows constructing a tunnel that shapes traffic from variety of sources. However, by focusing on a particular application such as web traffic, simpler version of this idea can be implemented at application level. For example similar defence can be embedded as a module on web browsers that could resize and pad web objects to the same size, or randomly re-order the web fetch requests. Implementation of such module is much easier but it is limited to one traffic source.

References

[1] Atkinson, J. S., Adetoye, O., Rio, M., Mitchell, J. E., and Matich, G. (2013). Your wifi is leaking: Inferring user behaviour, encryption irrelevant. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1097–1102.

[2] Atkinson, J. S., Rio, M., Mitchell, J. E., and Matich, G. (2014). Your wifi is leaking: Ignoring encryption, using histograms to remotely detect skype traffic. In *2014 IEEE Military Communications Conference*, pages 40–45.

[3] Back, A., Möller, U., and Stiglic, A. (2001). *Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems*, pages 245–257. Springer Berlin Heidelberg, Berlin, Heidelberg.

[4] Banchs, A., Serrano, P., and Oliver, H. (2007). Proportional Fair Throughput Allocation in Multirate IEEE 802.11e Wireless LANs. *Wirel. Netw.*, 13(5):649–662.

[5] Bissias, G. D., Liberatore, M., Jensen, D., and Levine, B. N. (2006). Privacy Vulnerabilities in Encrypted HTTP Streams. In Danezis, G. and Martin, D., editors, *Privacy Enhancing Technologies*, volume 3856 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg.

[6] Cai, X., Nithyanand, R., and Johnson, R. (2014a). Cs-buflo: A congestion sensitive website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 121–130, New York, NY, USA. ACM.

[7] Cai, X., Nithyanand, R., Wang, T., Johnson, R., and Goldberg, I. (2014b). A systematic approach to developing and evaluating website fingerprinting defenses. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 227–238, New York, NY, USA. ACM.

[8] Cai, X., Zhang, X. C., Joshi, B., and Johnson, R. (2012). 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. ACM.

[9] Callegati, F., Cerroni, W., and Ramilli, M. (2009). Man-in-the-middle attack to the https protocol. *IEEE Security and Privacy*, 7(1):78–81.

[10] Chaum, D. L. (1981). Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–90.

[11] Chen, S., Wang, R., Wang, X., and Zhang, K. (2010). Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *2010 IEEE Symposium on Security and Privacy*, pages 191–206.

[12] Cheng, H. and Avnur, R. (1998). Traffic analysis of ssl encrypted web browsing.

[13] Darroch, J. N. et al. (1964). On the Traffic-Light Queue. *The Annals of Mathematical Statistics*, 35(1):380–388.

[14] Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA. USENIX Association.

[15] Dubin, R., Dvir, A., Pele, O., and Hadar, O. (2016). I know what you saw last minute-encrypted http adaptive video streaming title classification. *arXiv preprint arXiv:1602.00490*.

[16] Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. (2012). 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.

[17] Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. (2013). Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 61–72, New York, NY, USA. ACM.

[18] Eryilmaz, A. and Srikant, R. (2005). Fair Resource Allocation in Wireless Networks Using Queue-length-based Scheduling and Congestion Control. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1794–1803. IEEE.

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

[20] Feghhi, S. and Leith, D. J. (2016a). A First-Hop Traffic Analysis Attack Against a Femtocell. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1060–1065.

[21] Feghhi, S. and Leith, D. J. (2016b). Time and place : Robustness of a traffic analysis attack against web traffic. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*.

[22] Feghhi, S. and Leith, D. J. (2016c). A web traffic analysis attack using only timing information. *IEEE Transactions on Information Forensics and Security*, 11(8):1747–1759.

[23] Feghhi, S., Leith, D. J., and Karzand, M. (2016). Proportional fair rate allocation for private shared networks. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 1006–1011.

[24] Flach, T., Dukkipati, N., Terzis, A., Raghavan, B., Cardwell, N., Cheng, Y., Jain, A., Hao, S., Katz-Bassett, E., and Govindan, R. (2013). Reducing web latency: the virtue of gentle aggression. *ACM SIGCOMM Computer Communication Review*, 43(4):159–170.

[25] Fung, B. C. M., Wang, K., Chen, R., and Yu, P. S. (2010). Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53.

[26] Ghaderi, J. and Srikant, R. (2010). Towards a theory of anonymous networking. In *2010 Proceedings IEEE INFOCOM*, pages 1–9.

[27] Ghaleb, T. A. (2016). Techniques and countermeasures of website/wireless traffic analysis and fingerprinting. *Cluster Computing*, 19(1):427–438.

[28] Gilad, Y. and Herzberg, A. (2012). Spying in the dark: Tcp and tor traffic analysis. In *Proceedings of the 12th International Conference on Privacy Enhancing Technologies*, PETS'12, pages 100–119, Berlin, Heidelberg. Springer-Verlag.

[29] Gong, X., Borisov, N., Kiyavash, N., and Schear, N. (2012). Website detection using remote traffic analysis. In *Privacy Enhancing Technologies*, pages 58–78. Springer.

[30] Gong, X., Kiyavash, N., and Borisov, N. (2010). 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. ACM.

[31] Guan, Y., Fu, X., Xuan, D., Shenoy, P., Bettati, R., and Zhao, W. (2001). NetCamo: Camouflaging Network Traffic for QoS-guaranteed Mission Critical Applications. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(4):253–265.

[32] Haataja, K. M. J. and Hypponen, K. (2008). Man-in-the-middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures. In *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*, pages 1096–1102.

[33] Hayes, J. and Danezis, G. (2016). k-fingerprinting: a robust scalable website fingerprinting technique. *arXiv preprint arXiv:1509.00789v3*.

[34] He, G., Yang, M., Gu, X., Luo, J., and Ma, Y. (2014). A novel active website fingerprinting attack against tor anonymous system. In *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*, pages 112–117.

[35] Herrmann, D., Wendolsky, R., and Federrath, H. (2009). 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. ACM.

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

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

[38] Jahani, H. and Jalili, S. (2016). A novel passive website fingerprinting attack on tor using fast fourier transform. *Computer Communications*, pages –.

[39] Juarez, M., Afroz, S., Acar, G., Diaz, C., and Greenstadt, R. (2014). A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 263–274, New York, NY, USA. ACM.

[40] Juarez, M., Imani, M., Perry, M., Diaz, C., and Wright, M. (2016). *Toward an Efficient Website Fingerprinting Defense*, pages 27–46. Springer International Publishing, Cham.

[41] Kadianakis, G. (2013). Packet Size Pluggable Transport and Traffic Morphing. Available at: https://research.torproject.org/techreports/morpher-2012-03-13.pdf.

[42] Kadloor, S. and Kiyavash, N. (2013). Delay Optimal Policies Offer Very Little Privacy. In *INFOCOM, 2013 Proceedings IEEE*, pages 2454–2462.

[43] Kennedy, D. P. (1972). The continuity of the single server queue. *Journal of Applied Probability*, 9(2):370–381.

[44] Keogh, E. J. and Pazzani, M. J. (2001). Derivative Dynamic Time Warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11.

[45] Kingman, J. F. C. (1962). On Queues in Heavy Traffic. *Journal of the Royal Statistical Society. Series B (Methodological)*, 24(2):383–392.

[46] Liberatore, M. and Levine, B. N. (2006). 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. ACM.

[47] Lu, L., Chang, E. C., and Chan, M. C. (2010). Website Fingerprinting and Identification Using Ordered Feature Sequences. In Gritzalis, D., Preneel, B., and Theoharidou, M., editors, *Computer Security – ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 199–214. Springer Berlin Heidelberg.

[48] Luo, X., Zhou, P., Chan, E. W. W., Lee, W., Chang, R. K. C., and Perdisci, R. (2011). HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows. In *In Proc. Network and Distributed Systems Symposium (NDSS). The Internet Society.*

[49] Malone, D., Kavanagh, D. F., and Murphy, N. R. (2013). Rogue femtocell owners: How mallory can monitor my devices. In *INFOCOM, 2013 Proceedings IEEE*, pages 3387–3392.

[50] Massoulie, L. and Roberts, J. (1999). Bandwidth Sharing: Objectives and Algorithms. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1395–1403 vol.3.

[51] Miller, B., Huang, L., Joseph, A. D., and Tygar, J. D. (2014). I know why you went to the clinic: Risks and realization of HTTPS traffic analysis. In De Cristofaro, E. and Murdoch, S. J., editors, *Privacy Enhancing Technologies*, volume 8555 of *Lecture Notes in Computer Science*, pages 143–163. Springer International Publishing.

[52] Mishra, A. (2015). Tradeoffs Between Anonymity and Quality of Services in Data Networking and Signaling Games.

[53] Mishra, A. and Venkitasubramaniam, P. (2012). Source Anonymity in Fair Scheduling: A Case for the Proportional Method. In *Communications (ICC), 2012 IEEE International Conference on*, pages 1118–1122. IEEE.

[54] Mohajeri Moghaddam, H., Li, B., Derakhshani, M., and Goldberg, I. (2012). Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 97–108, New York, NY, USA. ACM.

[55] Murdoch, S. J. and Danezis, G. (2005). Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pages 183–195, Washington, DC, USA. IEEE Computer Society.

[56] Nayak, G. N. and Samaddar, S. G. (2010). Different flavours of man-in-the-middle attack, consequences and feasible solutions. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 5, pages 491–495.

[57] Newell, G. F. (1960). Queues for a Fixed-Cycle Traffic Light. *Ann. Math. Statist.*, 31(3):589–597.

[58] Nithyanand, R., Cai, X., and Johnson, R. (2014). Glove: A bespoke website fingerprinting defense. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 131–134, New York, NY, USA. ACM.

[59] Panchenko, A., Lanze, F., Zinnen, A., Henze, M., Pennekamp, J., Wehrle, K., and Engel, T. (2016). Website fingerprinting at internet scale. In *Proceedings of the 23rd Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS 2016)*.

[60] Panchenko, A., Niessen, L., Zinnen, A., and Engel, T. (2011). 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. ACM.

[61] Ruffing, N., Zhu, Y., Libertini, R., Guan, Y., and Bettati, R. (2016). Smartphone reconnaissance: Operating system identification. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1086–1091.

[62] Schrittwieser, S., Frühwirt, P., Kieseberg, P., Leithner, M., Mulazzani, M., Huber, M., and Weippl, E. R. (2012). Guess who's texting you? evaluating the security of smartphone messaging applications. In *NDSS*. Citeseer.

[63] Schultz, A. M. (2016). Protecting consumer viewing habits: Reflections on the video privacy protection act.

[64] Shi, Y. and Biswas, S. (2015). Detecting tunneled video streams using traffic analysis. In *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–8.

[65] Song, D. X., Wagner, D., and Tian, X. (2001). Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001.

[66] Souders, S. (2009). Velocity and the bottom line. In *Velocity (Web Performance and Operations Conference)*.

[67] Sun, Q., Simon, D. R., Wang, Y.-M., Russell, W., Padmanabhan, V. N., and Qiu, L. (2002). Statistical identification of encrypted web browsing traffic. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 19–30.

[68] Thilakanathan, D., Chen, S., Nepal, S., Calvo, R., and Alem, L. (2014). A platform for secure monitoring and sharing of generic health data in the cloud. *Future Generation Computer Systems*, 35:102 – 113. Special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan.

[69] Tor Project: anonymity online (2016). https://www.torproject.org/.

[70] Tor Project: Users of Tor (2017). https://www.torproject.org/about/torusers.html.en.

[71] Valls, V. and Leith, D. (2015). Max-weight revisited: Sequences of nonconvex optimizations solving convex optimizations. *Networking, IEEE/ACM Transactions on*, PP(99):1–1.

[72] van den Broek, M. S., van Leeuwaarden, J. S. H., Adan, I. J. B. F., and Boxma, O. J. (2006). Bounds and Approximations for the Fixed-Cycle Traffic-Light Queue. *Transportation Science*, 40(4):484–496.

[73] Wagner, D., Schneier, B., et al. (1996). Analysis of the ssl 3.0 protocol. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, volume 1, pages 29–40.

[74] Wang, T., Cai, X., Nithyanand, R., Johnson, R., and Goldberg, I. (2014). Effective attacks and provable defenses for website fingerprinting. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 143–157, San Diego, CA. USENIX Association.

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

[76] Wang, T. and Goldberg, I. (2014). Comparing website fingerprinting attacks and defenses. Technical report.

[77] Wang, T. and Goldberg, I. (2015). Walkie-talkie: An effective and efficient defense against website fingerprinting. Technical report, Technical Report 2015-09, CACR, 2015. http://cacr. uwaterloo. ca/techreports/2015/cacr2015-09. pdf.

[78] Wang, X., Chen, S., and Jajodia, S. (2007). Network flow watermarking attack on low-latency anonymous communication systems. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 116–130.

[79] White, A. M., Matthews, A. R., Snow, K. Z., and Monrose, F. (2011). Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 3–18.

[80] Wikipedia (2017a). Computer network — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[81] Wikipedia (2017b). Internet protocol — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[82] Wikipedia (2017c). Internet protocol suite — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[83] Wikipedia (2017d). K-nearest neighbors algorithm — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[84] Wikipedia (2017e). Machine learning — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[85] Wikipedia (2017f). Naive bayes classifier — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[86] Wikipedia (2017g). Transmission control protocol — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[87] Wikipedia (2017h). Unsupervised learning — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[88] Wikipedia (2017i). User datagram protocol — wikipedia, the free encyclopedia. [Online; accessed 11-March-2017].

[89] Winter, P., Pulls, T., and Fuss, J. (2013). Scramblesuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 213–224, New York, NY, USA. ACM.

[90] Wright, C. V., Ballard, L., Coull, S. E., Monrose, F., and Masson, G. M. (2008). Spot me if you can: Uncovering spoken phrases in encrypted voip conversations. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 35–49.

[91] Wright, C. V., Coull, S. E., and Monrose, F. (2009). Traffic morphing: An efficient defense against statistical traffic analysis. In *In Proceedings of the 16th Network and Distributed Security Symposium*, pages 237–250. IEEE.

[92] Wu, T., Xue, Y., and Cui, Y. (2006). Preserving Traffic Privacy in Wireless Mesh Networks. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, WOWMOM '06, pages 459–461, Washington, DC, USA. IEEE Computer Society.

[93] Xue, Y. and Vasserman, E. Y. (2016). Simple and compact flow fingerprinting robust to transit through low-latency anonymous networks. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 765–773.

[94] yu, J. and Chan-Tin, E. (2014). Identifying webbrowsers in encrypted communications. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, WPES '14, pages 135–138, New York, NY, USA. ACM.

[95] Yu, S., Zhao, G., Dou, W., and James, S. (2012). Predicted packet padding for anonymous web browsing against traffic analysis attacks. *IEEE Transactions on Information Forensics and Security*, 7(4):1381–1393.

[96] Zhang, F., He, W., Liu, X., and Bridges, P. G. (2011). Inferring users' online activities through traffic analysis. In *Proceedings of the Fourth ACM Conference on Wireless Network Security*, WiSec '11, pages 59–70, New York, NY, USA. ACM.

[97] Zhang, L., Jia, W., Wen, S., and Yao, D. (2010). A man-in-the-middle attack on 3g-wlan interworking. In *Communications and Mobile Computing (CMC), 2010 International Conference on*, volume 1, pages 121–125.

[98] Zhou, W., Li, Q., Caesar, M., and Godfrey, P. (2011). ASAP: A low-latency transport layer. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 20. ACM.

Appendix A

Proof of Lemma 2

Proof. We begin by observing that,

$$\begin{aligned}
&\arg \min_{x\in[-1,1]} (\gamma - \lambda_g P)x \\
&= \arg \min_{x\in[-1,1]} (\gamma - \lambda_g P)(\bar{y}_g + \beta x) - \sum_{u\in\mathcal{U}} \bar{b}^{(u)} \qquad \text{(A.1)} \\
&= \arg \min_{x\in[-1,1]} L(\bar{y}_g + \beta x, \lambda_g) \qquad \text{(A.2)}
\end{aligned}$$

Now in (7.7) we then project $\bar{y}_g + \beta x$ onto interval $[0, y_{max}]$. Combining this projection, therefore, into the optimisation we have that steps (7.6)-(7.7) can be rewritten equivalently as

$$\bar{y}_{g+1} = \bar{y}_g + \beta \arg \min_{\substack{x\in[-1,1]: \\ \bar{y}_g+\beta x\in[0,y_{max}]}} L(\bar{y}_g + \beta x, \lambda_g) \qquad \text{(A.3)}$$

We consider two cases.

Case (i):

$$|L(\bar{y}_g + \beta x, \lambda_g) - L(\bar{y}_g, \lambda_g)| \geq \epsilon \qquad \text{(A.4)}$$

where $x \in \arg\min_{\substack{x\in[-1,1]: \\ \bar{y}_g+\beta x\in[0,y_{max}]}} L(\bar{y}_g + \beta x, \lambda_g)$. Note that since $(\gamma - \lambda_g P)\beta x \leq 0$ this case corresponds to $L(\bar{y}_g + \beta x, \lambda_g) - L(\bar{y}_g, \lambda_g) \leq -\epsilon$. Then,

$$\begin{aligned}
&L(\bar{y}_{g+1}, \lambda_{g+1}) - L(\bar{y}_g, \lambda_g) \\
&= L(\bar{y}_g + \beta x, \lambda_{g+1}) - L(\bar{y}_g, \lambda_g) && \text{(A.5)} \\
&= (\gamma - \lambda_{g+1}P)(\bar{y}_g + \beta x) - (\gamma - \lambda_g P)\bar{y}_g && \text{(A.6)} \\
&= (\gamma - \lambda_g P)\beta x + (\lambda_g - \lambda_{g+1})P(\bar{y}_g + \beta x) && \text{(A.7)} \\
&\leq -\epsilon + (\lambda_g - \lambda_{g+1})P(\bar{y}_g + \beta x) && \text{(A.8)}
\end{aligned}$$

Now, $|\lambda_{g+1} - \lambda_g| \leq \epsilon/(2Py_{max})$ and so

$$L(\bar{y}_{g+1}, \lambda_{g+1}) - L(\bar{y}_g, \lambda_g) \leq -\epsilon + \epsilon/2 \leq -\epsilon/2 \tag{A.9}$$

That is, the Lagrangrian is strictly decreasing.

Case (ii):

$$|L(\bar{y}_g + \beta x, \lambda_g) - L(\bar{y}_g, \lambda_g)| < \epsilon \tag{A.10}$$

Since $(\gamma - \lambda_g P)\beta x \leq 0$ this case corresponds to $-\epsilon < L(\bar{y}_g + \beta x, \lambda_g) - L(\bar{y}_g, \lambda_g) \leq 0$. Hence, $L(\bar{y}_{g+1}, \lambda_{g+1}) - L(\bar{y}_g, \lambda_g) = (\gamma - \lambda_g P)\beta x + (\lambda_g - \lambda_{g+1})P(\bar{y}_g + \beta x) \leq \epsilon/2$.

Thus,

$$\begin{aligned}
&L(\bar{y}_{g+1} + \beta x, \lambda_{g+1}) - L(\bar{y}_{g+1}, \lambda_{g+1}) \\
&\geq L(\bar{y}_{g+1} + \beta x, \lambda_{g+1}) - L(\bar{y}_g, \lambda_g) - \epsilon/2 && \text{(A.11)} \\
&= (\gamma - \lambda_{g+1}P)(\bar{y}_{g+1} + \beta x) - (\gamma - \lambda_g P)\bar{y}_g - \epsilon/2 && \text{(A.12)} \\
&= (\gamma - \lambda_g P)(\bar{y}_{g+1} - \bar{y}_g) + (\gamma - \lambda_g P)\beta x - \epsilon/2 \\
&\quad + (\lambda_g - \lambda_{g+1})P(\bar{y}_{g+1} + \beta x) && \text{(A.13)} \\
&\geq -\epsilon\beta - \epsilon\beta - \epsilon/2 - \epsilon/2 && \text{(A.14)} \\
&= -2\epsilon\beta - \epsilon && \text{(A.15)} \\
&= -(2\beta + 1)\epsilon' && \text{(A.16)}
\end{aligned}$$

Boundedness of Multiplier: $L(\bar{y}_g, \lambda_g)$ is strictly decreasing when $|L(\bar{y}_g + \beta x, \lambda_g) - L(\bar{y}_g, \lambda_g)| \geq \epsilon$ and once less than ϵ this difference increases to at most $(2\beta+1)\epsilon$. Hence, there exists a finite $\bar{g}$ such that for $g = \bar{g}$ then $-(2\beta+1)\epsilon < L(\bar{y}_g+\beta x, \lambda_g)-L(\bar{y}_g, \lambda_g) \leq 0$ and once this happens this inequality holds for all larger values of $g > \bar{g}$. Now

$L(\bar{y}_g + \beta x, \lambda_g) - L(\bar{y}_g, \lambda_g) = (\gamma - \lambda_g P)\beta x$ and so $-(2\beta + 1)\epsilon < (\gamma - \lambda_g P)\beta x \leq 0$. Provided $x \in \{-1, 1\}$ when $|L(\bar{y}_g + \beta x, \lambda_g) - L(\bar{y}_g, \lambda_g)| < (2\beta + 1)\epsilon$ (which we assume holds provided λ_{max} is sufficiently large) then it follows that $|\gamma - \lambda_g P| \leq (2\beta + 1)\epsilon/\beta$ for all $g \geq \bar{g}$. □