Hot or Not: Revealing Hidden Services by their Clock Skew

Steven J. Murdoch
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
http://www.cl.cam.ac.uk/users/sjm217/

ABSTRACT

Location-hidden services, as offered by anonymity systems such as Tor, allow servers to be operated under a pseudonym. As Tor is an overlay network, servers hosting hidden services are accessible both directly and over the anonymous channel. Traffic patterns through one channel have observable effects on the other, thus allowing a service's pseudonymous identity and IP address to be linked. One proposed solution to this vulnerability is for Tor nodes to provide fixed quality of service to each connection, regardless of other traffic, thus reducing capacity but resisting such interference attacks. However, even if each connection does not influence the others, total throughput would still affect the load on the CPU, and thus its heat output. Unfortunately for anonymity, the result of temperature on clock skew can be remotely detected through observing timestamps. This attack works because existing abstract models of anonymity-network nodes do not take into account the inevitable imperfections of the hardware they run on. Furthermore, we suggest the same technique could be exploited as a classical covert channel and can even provide geolocation.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Information Flow Controls*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Security, Experimentation

Keywords

Anonymity, Clock Skew, Covert Channels, Fingerprinting, Mix Networks, Temperature, Tor

CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.
Copyright 2006 ACM 1-59593-518-5/06/0010 ...$5.00.

1. INTRODUCTION

Hidden services allow access to resources without the operator's identity being revealed. Not only does this protect the owner, but also the resource, as anonymity can help prevent selective denial of service attacks (DoS) [35, 36]. Tor [15], has offered hidden services since 2004, allowing users to run a TCP server under a pseudonym. At the time of writing, there are around 80 publicly advertised hidden services, offering access to resources that include chat, low and high latency anonymous email, remote login (SSH and VNC), websites and even gopher. The full list of hidden services is only known by the three Tor directory servers.

Systems to allow anonymous and censorship-resistant content distribution have been desired for some time, but recently, anonymous publication has been brought to the fore by several cases of blogs being taken down and/or their authors being punished, whether imprisoned by the state [43] or being fired by their employers [5]. In addition to blogs, Tor hidden websites include dissident and anti-globalisation news, censored or otherwise controversial documents, and a PGP keyserver. It is clear that, given the political and legal situation in many countries, the need for anonymous publishing will remain for some time.

Because of the credible threat faced by anonymous content providers, it is important to evaluate the security, not only of deployed systems, but also proposed changes believed to enhance the security or usability. Guaranteed quality of service (QoS) is one such defence, designed to protect against indirect traffic-analysis attacks that estimate the speed of one flow by observing the performance of other flows through the same machine [33].

QoS acts as a countermeasure by preventing flows on an anonymity-network node from interfering with each other. However, an inevitable result is that when a flow is running at less than its reserved capacity, CPU load on the node will be reduced. This induces a temperature decrease, which affects the frequency of the crystal oscillator driving the system clock. We measure this effect remotely by requesting timestamps and deriving clock skew.

We have tested this vulnerability hypothesis using the current Tor implementation (0.1.1.16-rc), although – for reasons explained later – using a private instance of the network. Tor was selected due to its popularity, but also because it is well documented and amenable to study. However, the attacks we present here are applicable to the design of other anonymity systems, particularly overlay networks.

In Section 2 we review how hidden services have evolved from previous work on anonymity, discuss the threat models used in their design and summarise existing attacks. Then in Section 3 we provide some background on clock skew, the phenomenon we exploit to link hidden service pseudonyms to the server's real identity. In Section 4 we present the results of our experiments on Tor and discuss the potential impact and defences. Finally, in Section 5 we suggest how the general technique (of creating covert channels and side channels which cross between the digital and physical world) might be applied in other scenarios.

2. HIDDEN SERVICES

Low latency anonymity networks allow services to be accessed anonymously, in real time. The lack of intentional delay at first glance decreases security, but by increasing utility, the anonymity set can increase [1]. The first such proposal was the ISDN mix [39], but it was designed for a circuit switched network where all participants transmit at continuous and equal data rates and is not well suited to the more dynamic packet switched Internet. PipeNet [10] attempted to apply the techniques of ISDN mixes to the Internet, but while providing good anonymity guarantees, it is not practical for most purposes because when one node shuts down the entire network must stop; also, the cost of the dummy traffic required is prohibitive.

The Anonymizer [3] and the Java Anon Proxy (JAP) [7] provide low-latency anonymous web browsing. The main difference between them is that while Anonymizer is controlled by a single entity, traffic flowing through JAP goes through several nodes arranged in a fixed cascade. However, in neither case do they obscure where data enters and leaves the network, so they cannot easily support hidden services. This paper will instead concentrate on free-route networks, such as Freedom [4, 8] and the Onion Routing Project [42], of which Tor is the latest incarnation.

2.1 Tor

The attacks presented in this paper are independent of the underlying anonymity system and hidden service architecture, and should apply to any overlay network. While there are differing proposals for anonymity systems supporting hidden services, e.g. the PIP Network [17], Tor is a popular, deployed system, suitable for experimentation, so initially we will focus on it. Section 5 will suggest other cases where our technique can be used.

Tor hidden services are built on the connection anonymity primitive the network provides. As neither our attack nor the Tor hidden service protocol relies on the underlying implementation, we defer to [12, 13, 14, 15] for the full details. All that is important to appreciate in the remaining discussion is that Tor can anonymously tunnel a TCP stream to a specified IP address and port number. It does this by relaying traffic through randomly selected nodes, wrapping data in multiple layers of encryption to maintain unlinkability. Unlike email mixes, it does not intentionally introduce any delay: typical latencies are in the 10–100 ms range.

There are five special roles in a hidden service connection and all links between them are anonymised by the Tor network. The *client* wishes to access a resource offered by a *hidden server*. To do so, the client contacts a *directory server* requesting the address of an *introduction point*, which acts as an intermediary for initial setup. Then, both nodes connect to a *rendezvous point*, which relays data between the client and hidden server.

For clarity, some details have been omitted from this summary; a more complete description is in Øverlier and Syverson [38]. In the remainder of the paper, we will deal only with an established data connection, from the client to the rendezvous point and from there to the hidden server.

2.2 Threat Model

The primary goal of our attacker is to link a pseudonym (under which a hidden service is being offered) to the operator's real identity, either directly or through some intermediate step (e.g. a physical location or IP address). For the moment, we will assume that identifying the IP address is the goal, but Section 5.3 will discuss what else can be discovered, and some particular cases in which an IP address is hard to link to an identity.

Low-latency anonymity networks without dummy traffic, like Tor, cannot defend against a global passive adversary. Such an attacker simply observes inputs and outputs of the network and correlates their timing patterns, so called *traffic-analysis*. For the same reason, they cannot protect against traffic confirmation attacks, where an attacker has guessed who is communicating with whom and can snoop individual network links in order to validate this suspicion.

It is also common to assume that an attacker controls some of the anonymity network, but not all. In cases like Tor, which is run by volunteers subjected to limited vetting, this is a valid concern, and previous work has made use of this [33, 38]. However, the attacks we present here do not require control of any node, so will apply even to anonymity networks where the attacker controls no nodes at all.

In summary, we do not assume that our attacker is part of the anonymity network, but can access hidden services exposed by it. We do assume that he has a reasonably limited number of candidate hosts for the hidden service (say, a few thousand). However, we differ from the traffic confirmation case excluded above in that our attacker cannot observe, inject, delete or modify any network traffic, other than that to or from his own computer.

2.3 Existing Attacks

The first documented attack on hidden servers was by Øverlier and Syverson [38]. It proposes and experimentally confirms that a hidden service can be located within a few minutes to hours if the attacker controls one, or preferably two, network nodes. It relies on the fact that a Tor hidden server selects nodes at random to build connections. The attacker repeatedly connects to the hidden service, and eventually a node he controls will be the one closest to the hidden server. Now, by correlating input and output traffic, the attacker can confirm that this is the case, and so he has found the hidden server's IP address.

Another attack against Tor, but not hidden services *per se*, is described by Murdoch and Danezis [33]. The victim visits an attacker controlled website, which induces traffic patterns on the circuit protecting the client. Simultaneously, the attacker probes the latency of all Tor nodes and looks for correlations between the induced pattern and observed latencies. The full list of Tor nodes is, necessarily, available in the public directories along with their IP addresses. When there is a match, the attacker knows that the node is on the target circuit and so can reconstruct the path, although not discover the end node. In a threat model where the attacker has a limited number of candidates for the hidden service, this attack could also reveal its identity. Many hidden servers are also publicly advertised Tor nodes, in order to mask hidden server traffic with other Tor traffic, so this scenario is plausible.

Several defences are proposed by Murdoch and Danezis, which if feasible, should provide strong assurances against the attack. One is non-interference – where each stream going through a node is isolated from the others. Here each Tor node has a given capacity, which is divided into several slots. Each circuit is assigned one slot and is given a guaranteed data rate, regardless of the others.

Our new observation, which underpins the attack presented, is that when circuits carried by a node become idle, its CPU will be less active, and so cool down. Temperature has a measurable effect on clock skew, and this can be observed remotely. We will show that an attacker can thus distinguish between a CPU in idle state and one that is busy. But first some background is required.

3. CLOCK SKEW AND TEMPERATURE

Kohno *et al.* [24] used timing information from a remote computer to fingerprint its physical identity. By examining timestamps from the machine they estimated its *clock skew*: the ratio between actual and nominal clock frequencies.

They found that a particular machine's clock skew deviates very little over time, around 1–2 parts per million (ppm), depending on operating system, but that there was a significant difference between the clock skews (up to 50 ppm) of different machines, even identical models. This allows a host's clock skew to act as a fingerprint, linking repeated observations of timing information. The paper estimates that, assuming a stability of 1 ppm, 4–6 bits of information on the host's identity can be extracted.

Two sources of timestamps were investigated by Kohno *et al.*: ICMP timestamp requests [40] and TCP timestamp options [21]. The former has the advantage of being of a fixed nominal frequency (1 kHz), but if a host is Network Time Protocol (NTP) [28] synchronised, the ICMP timestamp was found to be generated after skew adjustment, so defeating the fingerprinting attack. The nominal frequency of TCP timestamps depends on the operating system, and varies from 2 Hz (OpenBSD 3.5) to 1 kHz (Linux 2.6.11). However, it was found to be generated before NTP correction, so attacks relying on this source will work regardless of the NTP configuration. Additionally, in the special case of Linux, Murdoch and Lewis [34] showed how to extract timestamps from TCP sequence numbers.

We will primarily use TCP timestamps, which are enabled by default on most modern operating systems. They improve performance by providing better estimates of round trip times and protect against wrapped sequence numbers on fast networks. Because of their utility, TCP timestamps are commonly passed by firewalls, unlike ICMP packets and IP options, so are widely applicable. The alternative measurement techniques will be revisited in Section 5.4.

3.1 Background and Definitions

Let $T(t_\mathrm{s})$ be the timestamp sent at time t_s. Unless specified otherwise, all times are relative to the receiver clock. As we are interested in changes of clock frequency, we split skew into two components, the constant s_c and the time-varying part $s(t)$. Without loss of generality, we assume that the time-varying component is always negative.

Before a timestamp is sent, the internal value of time is converted to a number of *ticks* and rounded down. The nominal length of a tick is the clock's resolution and the reciprocal of this is its nominal frequency, h. The relationship between the timestamp and input parameters is thus:

$$T(t_\mathrm{s}) = \left\lfloor h \cdot \left(t_\mathrm{s} + s_\mathrm{c} t_\mathrm{s} + \int_0^{t_\mathrm{s}} s(t)\,dt\right)\right\rfloor \tag{1}$$

Now we sample timestamps T_i sent at times t_{s_i} chosen uniformly at random between consecutive ticks, for all i in $[1 \ldots n]$, with $t_{\mathrm{s}_1} = 0$. The quantisation noise caused by the rounding can be modelled as subtracting a random variable c with uniform distribution over the range $[0, 1)$. Also, by dividing by h, we can recover the time according to the sender in sample i:

$$\tilde{t}_i = T_i/h = t_{\mathrm{s}_i} + s_\mathrm{c} t_{\mathrm{s}_i} + \int_0^{t_{\mathrm{s}_i}} s(t)\,dt - c_i/h \tag{2}$$

We cannot directly measure the clock skew of a remote machine, but we can calculate the *offset*. This is the difference between a clock's notion of the time and that defined by the reference clock (receiver). The offset o_i can be found by subtracting t_{s_i} from $\tilde{t}_i$. However, the receiver only knows the time t_{r_i} when a packet was received.

Let d_i be the latency of a packet, from when it is timestamped to when it is received, then $t_{\mathrm{s}_i} = t_{\mathrm{r}_i} - d_i$. Skew is typically small (< 50 ppm) so the effect of latency to these terms will be dominated by the direct appearance of d_i and is ignored otherwise. The measured offset is thus:

$$\tilde{o}_i = \tilde{t}_i - t_{\mathrm{r}_i} = s_\mathrm{c} t_{\mathrm{r}_i} + \int_0^{t_{\mathrm{r}_i}} s(t)\,dt - c_i/h - d_i \tag{3}$$

Figure 1 shows a plot of the measured offset over packet receipt time. Were the sampling noise c/h, latency introduced noise d and variable skew $s(t)$ absent, the constant skew s_c would be the derivative of measured offset with respect to time. To form an estimate of the constant skew observed, $\hat{s}_\mathrm{c}$, in the presence of noise, we would like to remove these terms. Note that in (3) the noise contributions, as well as $s(t)$, are both negative.

Following the approach of Kohno *et al.*, we remove the terms by fitting a line above all measurements while minimising the mean distance between each measurement and the point on the line above it. By applying the linear-programming based algorithm described by Moon *et al.* [29], we derive such a line. More formally this finds an estimate of the linear offset component $\hat{o}(t) = \hat{s}_\mathrm{c} \cdot t + \beta$ such that, for all samples, $\hat{o}(t_{\mathrm{r}_i}) > \tilde{o}_i$ and minimises the expression:

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \left(\hat{o}(t_{\mathrm{r}_i}) - \tilde{o}_i\right) \tag{4}$$

The offset $\hat{o}(t)$ is also plotted on Figure 1. The band of offset samples below the line is due to the sampling noise c/h, as illustrated by the different width depending on h. Points are outside this band because of jitter in the network delay (any constant component will be eliminated), but latencies are tightly clustered below a minimum which remains fixed during the test. This is to be expected for an uncongested network where routes change rarely. The characteristics of these noise sources will be discussed further in Section 5.4.

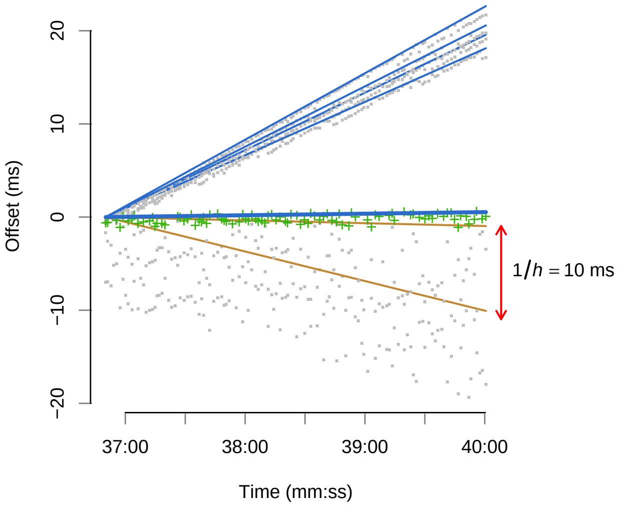

Figure 1: Offset between TCP timestamps of seven machines and the measurer's clock over time. The bottom two lines (—) show clocks with 100 Hz resolution and the others are 1 kHz. The range of the quantisation noise is $[0, 1/h)$, as indicated for the $h = 100$ Hz case. The time since the beginning of the experiment was subtracted from the measurer's clock and the first timestamp received was subtracted from all timestamps. All machines were on the same LAN, except one (+) which was accessed over a transatlantic link, through 14 hops.

3.2 Impact of Temperature

The effect of temperature on remote clock skew measurements has been well known to the NTP community since the early 1990s [25, 27] and was mentioned by Kohno *et al.* However, we believe our paper to be the first that proposes *inducing* temperature change and measuring the change in clock skew, in order to attack an anonymity system.

As shown in Figure 2, the frequency of clock crystals varies depending on temperature [9]. Exactly how depends on tradeoffs made during manufacture. The figure shows an AT-cut crystal common for PCs, whose skew is defined by a cubic function. BT-cut is more common for sub-megahertz crystals and is defined by a quadratic. The angle of cut alters the temperature response and some options are shown. It can be seen that improving accuracy in the temperature range between the two turning points degrades performance outside these values. Over the range of temperatures encountered in our experiments, skew response to temperature is almost linear, so for simplicity we will treat it as such.

The linear offset fit shown in Figure 1 matches almost perfectly, excluding noise. This indicates that although temperature varied during the sample period, the constant skew s_{c} dominates any temperature dependence $s(t)$ present. Nevertheless, the temperature dependent term $s(t)$ is present and is shown in Figure 3. Here, $\hat{o}(t_{\mathrm{r}_i})$ has been subtracted from all $\tilde{o}_i$, removing our estimate of constant skew $\hat{s}_{\mathrm{c}}$. To estimate the variable skew component $\hat{s}(t)$, the resulting offset is differentiated, after performing a sliding window line-fitting. We see that as the temperature in the room varied over the day, there is a correlated change in clock skew.

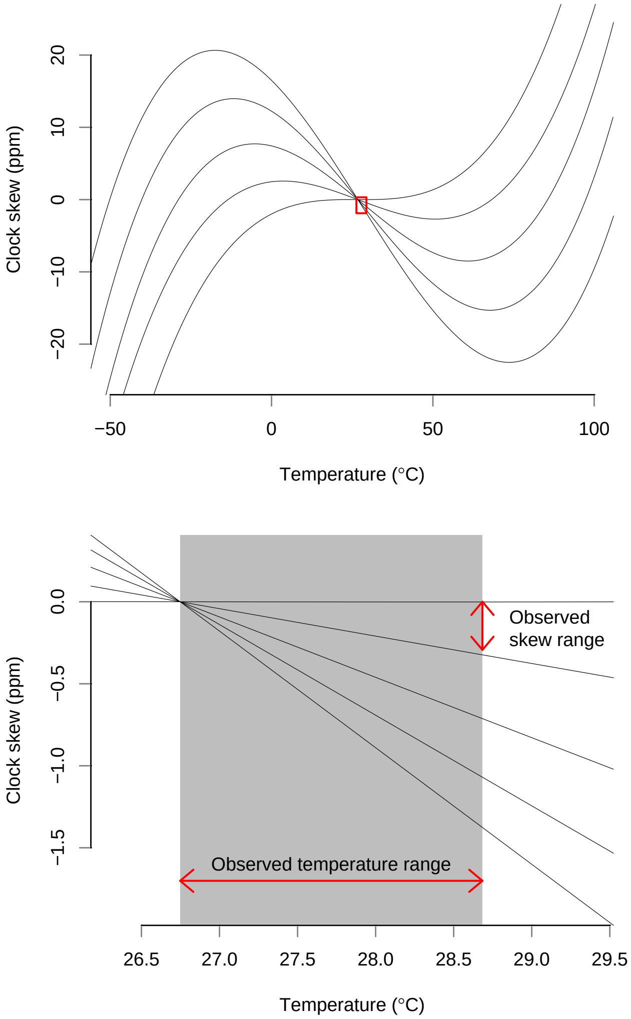

Figure 2: AT-cut crystal clock skew over two temperature ranges. Full operational range is shown above, with indicated zoomed-in area below. On the zoomed graph, the temperature and skew ranges found in Figure 5(a) are shown. As skews are relative, the curves have been shifted vertically so that skew is zero at the minimum observed temperature.

4. ATTACKING TOR

We aim to show that a hidden server will exhibit measurably different behaviour when a particular connection is active compared to when it is idle. Where the Murdoch and Danezis attacks [33] probed the latency of other connections going through the same node, we measure clock skew.

This is because when a connection is idle, the host will not be performing as many computations and so cool down. The temperature change will affect clock skew, and this can be observed remotely by requesting timestamps. The goal of our experiment is to verify this hypothesis.

Such an attack could be deployed in practice by an attacker using one machine to access the hidden service, varying traffic over time to cause the server to heat up or cool down. Simultaneously, he probes all candidate machines for

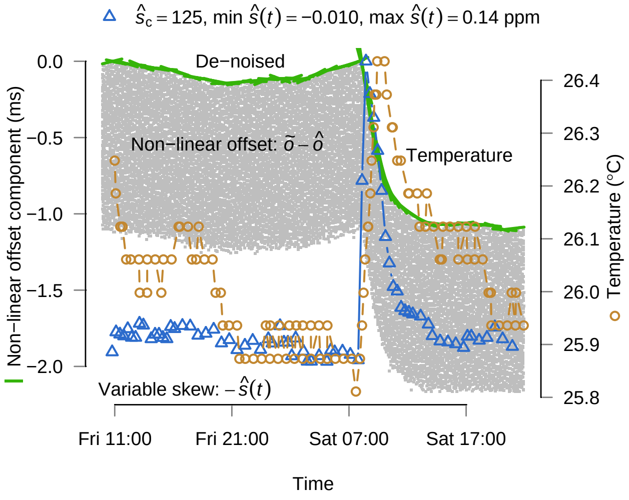

Figure 3: Offset after removing linear component (i.e. $\tilde{o} - \hat{o}$). The line (—) above is the de-noised version. The △ show the negated slope of each piece ($-\hat{s}(t)$) and ○ show the temperature. The maximum and minimum values of $\hat{s}(t)$ are shown, along with the constant skew $\hat{s}_c$. Results are from a minitower PC with ASUS A7M266 motherboard and 1.3 GHz AMD Athlon processor.

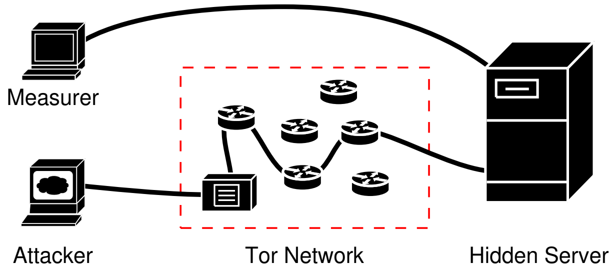

Figure 4: Experimental setup with four computers.

timestamps. From these the attacker infers clock skew estimates and when a correlation between skew and the induced load pattern is found, the hidden service is de-anonymised.

The reliability and performance of the Tor network for hidden servers is currently quite poor, so, to simplify obtaining results, our experiments were run on a private Tor network. We see no reason these results would not transfer to the real Tor network, even when it is made more reliable and resistant to the Murdoch and Danezis attacks.

The computers used in each test (shown in Figure 4) are:

Hidden Server: Tor client and webserver, hosting a 10 MB file; fitted with a temperature sensor.

Tor Network: Two Tor directory server processes and five normal servers, which can act as introduction and rendezvous points, all unmodified. While all processes are on the same machine, this does not invalidate our results as only the Hidden Server is being analysed.

Attacker: Runs the Tor client, repeatedly requesting the file hosted by Hidden Server, through the Tor Network. For performance, this is modified to connect directly to the rendezvous point.

Measurer: Connects directly to the Hidden Server's public IP address, requesting TCP timestamps, ICMP timestamps and TCP sequence numbers, although only the results for the first are shown.

For two hours the 10 MB file is repeatedly downloaded over the Tor network, with up to 10 requests proceeding in parallel. Then for another two hours no requests are made. During both periods timestamps are requested directly from the server hosting the hidden service at intervals of 1 s plus a random period between 0 s and 1 s. This is done to meet the assumption of Section 3.1, that samples are taken at random points during each tick. Otherwise, aliasing artifacts would be present in the results, perturbing the line-fitting.

Finally, the timestamps are processed as described in Section 3.2. That is, estimating the constant skew through the linear programming algorithm and removing it; dividing the trace into pieces and applying the linear-programming algorithm a second time to estimate the varying skew.

Were an attacker to deploy this attack, the next step would be to compare the clock skew measurements of all candidate servers with the load pattern induced on the hidden service. To avoid false-positives, multiple measurements are needed. The approach taken by Murdoch and Danezis [33] is to treat the transmission of the load pattern as a covert channel and send a pseudorandom binary sequence. Thus, after n bits are received, the probability of a false-positive is 2^{-n}. From inspection, we estimate the capacity of the covert channel to be around 2–8 bits per hour. An alternative taken by Fu *et al.* [16] is to induce a periodic load pattern which can be identified in the power spectrum of the Fourier transformed clock skew measurements. With either approach, the confidence could be increased to arbitrary levels by running the attack for longer.

4.1 Results

Overall throughput was limited by the CPU of the server hosting the private Tor network, so the fastest Hidden Server tested ran at around 70% CPU usage while requests were being made. CPU load on the Hidden Server was almost all due to the Tor process, we suspect due to it performing cryptographic operations. A 1–1.5 °C temperature difference was induced by this load modulation.

Ideally, the measuring machine would have a very accurate clock, to allow comparison of results between different experiments over time and with different equipment. This was not available for these experiments, however as we are interested only in relative skews, only a stable clock is needed, for which a normal PC sufficed. It would also be desirable to timestamp packets as near as possible to receipt, so while adding the timestamp at the network card would be preferable, the one inserted by the Linux kernel and exposed through the `pcap` [22] interface has proved to be adequate.

Figure 5 shows the results of two experimental runs, in the same style as Figure 3. Note that the top graph shows a relationship between clock skew and temperature opposite to expectations; namely when temperature increases, the clock has sped up. One possible explanation is that the PC is using a temperature compensated crystal oscillator (TCXO),

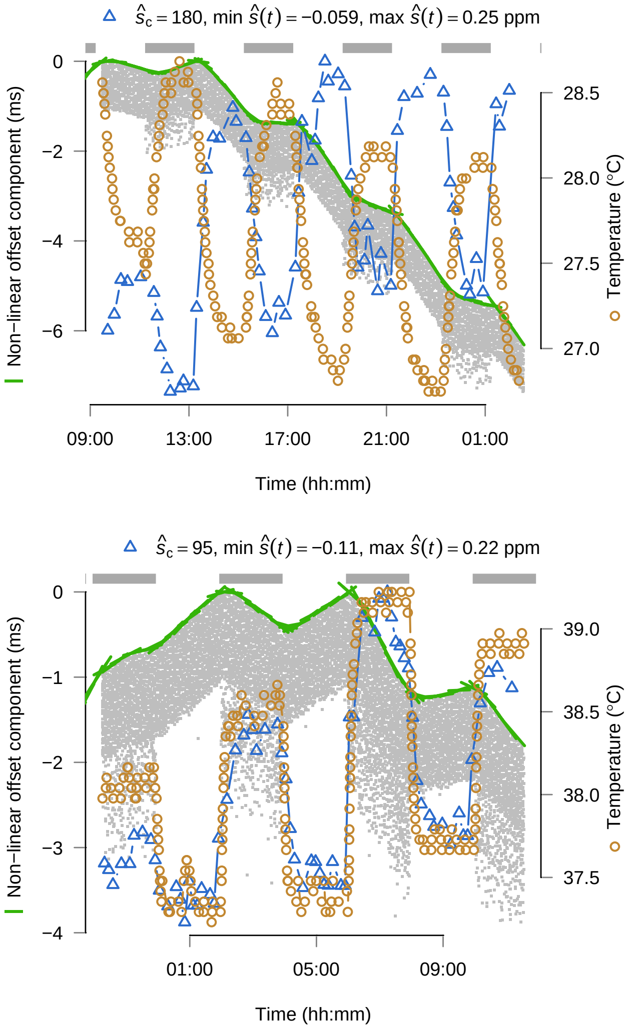

Figure 5: Clock skew measurements for two machines. The graph is as Figure 3, but the grey bars at the top indicate when the hidden server was being exercised. The top graph is from a mini-tower PC with Dell GX1MT motherboard and Intel Pentium II 400 MHz processor; the bottom is from a mini-tower PC with ASUS A7V133 motherboard and AMD Athlon 1.2 GHz processor.

but is over compensating; another is that the temperature curve for the crystal is different from Figure 2. In both cases there is a clear correlation between temperature and skew, despite only a modest temperature change.

While the CPU is under load, there is increased noise present in the results. This could be due to increased latency on the network, or more likely because the CPU is so busy, the operating system sometimes allocates a quantum to the Tor process in between adding a timestamp to a packet and dispatching it. However, note that the minimum latency is unchanged (and is often reached) so the linear programming algorithm still performs well. Were the minimum to change, then a step in the graph would be expected, rather than the smooth curve apparent.

4.2 Discussion

Murdoch and Danezis [33] proposed a defence to their flow interference attacks, that did not require dummy traffic. It was to ensure that no anonymous stream flowing through a hidden server should affect any other, whether it belonged to the anonymity service or not. All connections are thus given a fixed quality of service, and if the maximum number of connections is reached, further connections are refused.

Implementing this is non-trivial as QoS must not only be guaranteed by the host (e.g. CPU resources), but by its network too. Also, the impact on performance would likely be substantial, as many connections will spend much of their time idle. Whereas currently the idle time would be given to other streams, now the host carrying such a stream cannot reallocate any resources, thus opening a DoS vulnerability. However, there may be some suitable compromise, for example dynamic limits which change sufficiently slowly that they leak little information.

Even if such a defence were in place, our temperature attacks would still be effective. While changes in one network connection will not affect any other connections, clock skew is altered. This is because the CPU will remain idle during the slot allocated to a connection without pending data. Unless steps are taken to defend against our attacks, the reduced CPU load will lower temperature and hence affect clock skew. To stabilise temperature, computers could be modified to use expensive oven controlled crystal oscillators (OCXO), or always run at maximum CPU load. External access to timing information could be restricted or jittered, but unless all incoming connections were blocked, extensive changes would be required to hide low level information such as packet emission triggered by timer interrupts.

While the above experiments were on Tor, we stress that our techniques apply to any system that hides load through maintaining QoS guarantees. Also, there is no need for the anonymity service to be the cause of the load. For example, Dean and Stubblefield [11] show that because SSL allows the client to force a server to perform an RSA operation before doing any substantial work itself, DoS attacks can be mounted well before the connection is saturated. Such techniques could be used to attack hidden servers where the anonymity network cannot sustain high throughput.

Inducing clock skew and remotely measuring it can be seen as a *thermal covert channel* because attacking a hidden server could be modelled as violating an information flow control policy in a distributed system. The client accessing the hidden service over the anonymity network is using the link between between the server's pseudonym and its public IP address, which is information at a "high" confidentiality level. However, the client is prevented from leaking this information by the trusted computing base of the anonymity network. The user accessing the hidden server directly only has access to "low" information, the real IP address by itself, however if the "high" process can leak information to the "low" process, the server's anonymity is violated.

This scenario is analogous to covert channel attacks on the $*$-property of the BLP model [6]: that processes must not be able write to a process lower than its own privilege level. This approach to the analysis of anonymity systems was proposed by Moskowitz *et al.* [30, 31, 37], but we have shown here further evidence that past research in the field of covert channels can be usefully applied in enhancing the security of modern-day anonymity systems.

5. EXTENSIONS AND FUTURE WORK

The above experiments presented an example of how temperature induced clock skew can be a security risk, but we believe that this is a more general, and previously underexamined, technique which can be applied in other situations. In this section we shall explore some of these cases and propose some future directions for research.

5.1 Classical Covert Channels

The above section discussed an unconventional application of covert channels, that is within a distributed system where users can only send data but not execute arbitrary code. However, clock skew can also be used in conventional covert channels, where an operating system prevents two processes communicating which are on the same computer and can run arbitrary software.

CPU load channels have been extensively studied in the context of multilevel secure systems. Here, two processes share CPU time but the information flow control policy prohibits them from directly communicating. Each can still observe how much processing time it is getting, thus inferring the CPU usage of the other.

A process can thus signal to another by modulating load to encode information [26]. One defence against this attack is to distort the notion of time available to processes [19] but another is fixed scheduling and variations, ensuring that the CPU of one process cannot interfere with the resources of any at a conflicting security rating [20]. Temperature induced clock skew can circumvent the latter countermeasure. Covert channels are also relevant to recently deployed separation kernels such as MILS [2, 44].

Figure 6 shows one such example. In previous cases, the temperature in the measured machine has been modulated, but now we affect the clock skew of the measurer. This graph was plotted in the same way as before, but on the measurer machine the `CPUBurn` program [41] was used to induce load modulation, affecting the temperature as shown. Timestamps are collected from a remote machine and as we are calculating relative clock skew, we see the inverse of the measurer's clock skew, assuming the remote clock is stable.

Note that the temperature difference is greater than before (5 °C vs. 1–1.5 °C). This is because we are no longer constrained by the capacity of the Tor network, and can optimise our procedure to induce the maximum temperature differential. While this attack is effective, it requires fairly free access to network resources, which is not common in the general case of high-assurance systems where covert channels are a serious concern.

Where access to a remote timing source is blocked, the skew between multiple clock crystals within the same machine, due to their differing temperature responses and proximity to the heat source, could be used. For example, in a typical PC, the sound card has a separate crystal from the system clock. A process could time how long it takes (according to the system clock), to record a given number of samples from the sound card, thus estimating the skew between the two crystals.

5.2 Cross Computer Communication

Physical properties of shared hardware have previously been proposed as a method of creating covert channels. For example, hard disk seek time can be used to infer the previous position of the disk arm, which could have been affected by "high" processes [23]. However, with temperature, such effects can extend across "air-gap" security boundaries.

Our experiments so far have not shown evidence of one desktop computer being able to induce a significant temperature change in another which is in the same room, but the same may not be true of rack-mount machines. Here, a 3 °C temperature change in a rack-mount PC has been induced by increasing load on a neighbouring machine [18]. Blade servers, where multiple otherwise independent servers are mounted as cards in the same case, sharing ventilation and power, have even more potential for thermal coupling.

If two of these cards are processing data at different security levels, the tight environmental coupling could lead to a covert channel as above, even without the co-operation of the "low" card. For example, if a "low" webserver is hosted next to a "high" cryptographic co-processor which does not have Internet access, the latter could leak information to an external adversary by modulating temperature while the webserver clock-skew is measured. Side-channels are also conceivable, where someone probing one card could estimate the load of its siblings.

We simulated this case by periodically (2 hours on, 2 hours off) exposing a PC to an external heat source while a second computer measured the clock skew. The results showed that 3 °C temperature changes can be remotely received. Additionally, this confirmed that it is temperature causing the observed clock skew in the previous experiments, and not an OS scheduling artifact. The resulting graph was similar to Figure 5 except there is no increased noise during heating, as would be expected from the hypothesised interference attack resistant anonymity system.

5.3 Geolocation

In the attacks on anonymity systems so far, we have been inducing load through the anonymity system and measuring clock skew directly. An alternative is to measure clock skew through the anonymity network and let the environment alter the clock skew. This allows an attacker to observe temperature changes of a hidden server, so infer its location.

Clock skew does not allow measurement of absolute temperature, only changes. Nevertheless this still could be adequate for geolocation. Longitude could simply be found by finding the daily peak temperature to establish local time. To find latitude, the change in day length over a reasonably long period could be used.

It was apparent in our experiments when a door to the cooler corridor was left open, so national holidays or when daylight saving time comes into effect might be evident. Distortion caused by air-conditioning could be removed by inferring the temperature from the duty cycle (time on vs. time off) of thermostatically controlled appliances.

In this section we have assumed that we probe through the anonymity network. In the case of Tor, this will introduce significant jitter, and it is unclear how badly this will affect timing measurements. Alternatively, the attacker could connect directly to the external IP address.

This raises the question of utility – often IP addresses can easily be mapped to locations [32]. However, this is not always the case. For example, IP anycast and satellite connections are hard to track to a location; as are users who seek to hide by using long-distance dialup. While latency in the last two cases is high, jitter can be very low, lending itself to the clock skew attacks.

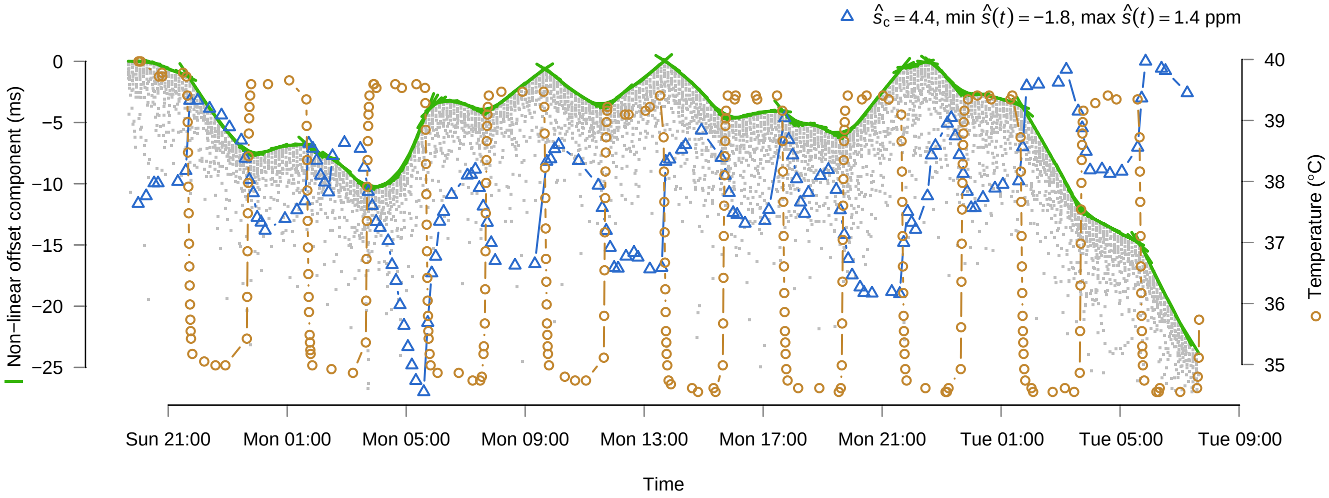

Figure 6: Clock skew measurements of a remote machine while modulating CPU load of the measurer (mini-tower, Intel D875 motherboard, Pentium 4 3.2 GHz CPU), for which temperature is also shown. The measurer and remote machine are separated by a transatlantic link, so the noise level is higher.

5.4 Noise Sources and Mitigation

In the above section, we proposed acquiring timing information from a hidden server through the anonymity network. Here, in addition to the problem of increased jitter, the timing sources we have used (ICMP/TCP timestamps and TCP sequence numbers) may not be available. For example, Tor operates at the TCP layer so these possibilities do not exist, whereas Freedom [4, 8] allows the transmission of arbitrary IP packets.

One option proposed by Kohno *et al.* is to use a Fourier transform to extract the frequency of a periodic event, for example, packet emission caused by a timer interrupt. Another possibility is to use application level timestamps. The most common Tor hidden service is a web server, typically using Apache, and by default this inserts a timestamp into HTTP headers. However, this only has a 1 Hz resolution, compared to the 1 kHz used in our experiments.

To improve performance in these adverse conditions by mitigating the effect of noise, we must first understand the source. The noise component of (3) is the sum of two independent parameters: quantisation noise c_i/h and latency d_i, although we only care about the variable component of the latter, *jitter* j_i. The quantisation noise is chosen uniformly at random from $[0, 1/h)$, and so is trivially modelled, but j_i can only be found experimentally.

The top graph of Figure 7 shows the smoothed probability density for round trip jitter (divided by two), which can be measured directly. If we assume that forward and return paths have independent and similar jitter, then j_i would be the same distribution. By convolving the estimated densities of the two noise sources, we can show the probability density of the sum, which matches the noise measurements of clock offset shown on the bottom of Figure 7.

The linear programming algorithm used for skew calculations is effective at removing j_i, because values are strongly skewed towards the minimum, but for c_i/h, it is possible to do better. One obvious technique is to increase h by selecting a higher resolution time source. We have used TCP timestamps in this paper, primarily with Linux 2.6, which have a nominal frequency of 1 kHz. Linux 2.4 has a 100 Hz TCP timestamp clock, so for this, ICMP timestamps may be a better option, as they increment at a nominal 1 kHz.

Unlike TCP timestamps, we found ICMP to be affected by NTP, but initial experiments show that while this is a problem for finding out absolute skew, the NTP controlled feedback loop in Linux intentionally does not react fast enough to hide the changes in skew this paper considers. Another option with Linux is to use TCP sequence numbers, which are the sum of a cryptographic result and a 1 MHz clock. Over short periods, the high h gives good results, but as the cryptographic function is re-keyed every 5 minutes, maintaining long term clock skew figures is non-trivial.

Note that to derive (2) from (1) we assumed that samples are taken at random points between ticks. This allows the floor operation ($\lfloor\,\rfloor$) to be modelled as uniformly distributed noise. Regular sampling introduces aliasing artifacts which interfere with the linear programming algorithm.

However, the points which contribute to the accuracy of the skew estimate, those near the top of the noise band, are from timestamps generated just after a tick. Here, the value of c_i is close to zero, and just before the tick, c_i is close to one and the timestamp is one less. An attacker could use the previous estimate of skew to target this point and identify which side of the transition the sample was taken. From this, he can estimate when the tick occurred and so refine the skew estimate.

This approach effectively removes the quantisation error. Rather than $1/h$ defining the noise band, it now only limits the sampling rate to h. Multiple measurements would still be needed to remove jitter, most likely by using the same linear programming algorithm as in the simple case, but perhaps also taking into consideration the round trip time. Adequate results can be achieved using naïve random sampling, but the improved technique would be particularly valuable for low resolution clocks, such as the 1 Hz Apache HTTP timestamp mentioned above.

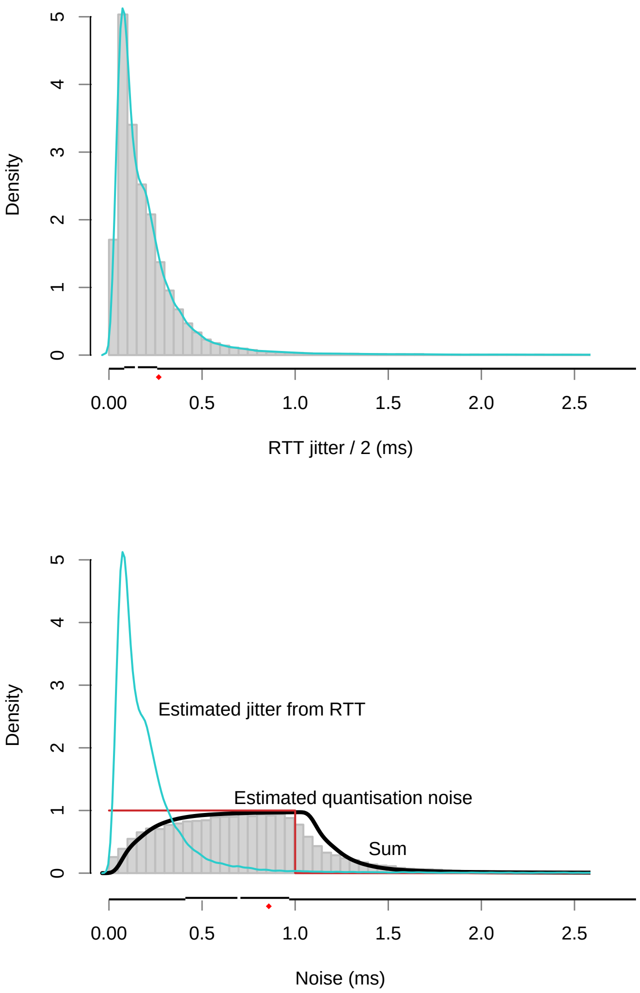

Figure 7: Top graph shows probability density of measured round trip time jitter (divided by two) with overlaid kernel density estimate (—). Bottom graph is density of measured offset noise, overlaid with the above density, uniform quantisation noise model (—) and the calculated sum of the two components (—**). The breaks in the x axis indicate quartiles and the mean is shown as ◆. Measurements were taken over a transatlantic link (14 hops).**

6. CONCLUSION

We have shown that changes in clock skew, resulting from only modest changes in temperature, can be remotely detected even over tens of router hops. Our experiments show that environmental changes, as well as CPU load, can be inferred through these techniques. However, our primary contribution is to introduce an attack whereby CPU load induced through one communication channel affects clock skew measured through another. This can link a pseudonym to a real identity, even against a system that ensures perfect non-interference when considered in the abstract.

We have demonstrated how such attacks could be used against hidden services. We validated our expectations by testing them with the deployed Tor code, not simulations, although on a private network, rather than the publicly accessible one. Our results show that proposed defences against interference attacks which use quality of service guarantees, are not as effective as previously thought. We suggest that when designing such systems, considering only the abstract operating system behaviour is inadequate as their implementation on real hardware can substantially decrease security.

We proposed future directions for security research using thermal covert channels. These include allowing two computers which share the same environment, but are otherwise isolated, to communicate. Also, processes on the same computer, under an information-flow-control policy, can send information through temperature modulation, despite fixed scheduling preventing CPU load based covert channels.

Finally, we proposed how localised temperature changes might aid geolocation and suggested methods to deal with low resolution clocks.

7. ACKNOWLEDGEMENTS

Thanks are due to Richard Clayton and Roger Dingledine for assistance in running the experiments. We also thank Markus Kuhn, Nick Mathewson, Lasse Øverlier, and the anonymous reviewers for their valuable comments.

8. REFERENCES

[1] A. Acquisti, R. Dingledine, and P. F. Syverson. On the economics of anonymity. In R. N. Wright, editor, *Financial Cryptography*, volume 2742 of *LNCS*, pages 84–102. Springer-Verlag, 2003.

[2] J. Alves-Foss, C. Taylor, and P. Omanl. A multi-layered approach to security in high assurance systems. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, Hawaii, January 2004. IEEE CS.

[3] Anonymizer, Inc. `http://www.anonymizer.com/`.

[4] A. Back, I. Goldberg, and A. Shostack. Freedom Systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.

[5] BBC News. US blogger fired by her airline, November 2004. `http://news.bbc.co.uk/1/technology/3974081.stm`.

[6] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations. Technical Report 2547, Volume I, MITRE Corporation, March 1973.

[7] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 115–129. Springer-Verlag, July 2000.

[8] P. Boucher, A. Shostack, and I. Goldberg. Freedom Systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

[9] C-MAC MicroTechnology. HC49/4H SMX crystals datasheet, September 2004. `http://www.cmac.com/mt/databook/crystals/smd/hc49_4h_smx.pdf`.

[10] W. Dai. PipeNet 1.1, November 1998. `http://www.eskimo.com/~weidai/pipenet.txt`.

[11] D. Dean and A. Stubblefield. Using client puzzles to protect TLS. In *Proceedings of the 10th USENIX Security Symposium*, Aug. 2001.

[12] R. Dingledine and N. Mathewson. Tor protocol specification. Technical report, The Free Haven Project, October 2004. `http://tor.eff.org/cvs/doc/tor-spec.txt`.

[13] R. Dingledine and N. Mathewson. Tor path specification. Technical report, The Free Haven Project, April 2006. `http://tor.eff.org/cvs/doc/path-spec.txt`.

[14] R. Dingledine and N. Mathewson. Tor rendezvous specification. Technical report, The Free Haven Project, February 2006. `http://tor.eff.org/cvs/doc/rend-spec.txt`.

[15] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[16] X. Fu, Y. Zhu, B. Graham, R. Bettati, and W. Zhao. On flow marking attacks in wireless anonymous communication networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 493–503, Columbus, Ohio, USA, June 2005. IEEE CS.

[17] I. Goldberg. *A Pseudonymous Communications Infrastructure for the Internet*. PhD thesis, UC Berkeley, December 2000.

[18] H. Grundy. Personal communication.

[19] W.-M. Hu. Reducing timing channels with fuzzy time. In *1991 IEEE Symposium on Security and Privacy*, pages 8–20, Oakland, California, May 1991. IEEE CS.

[20] W.-M. Hu. Lattice scheduling and covert channels. In *1992 IEEE Symposium on Security and Privacy*, pages 52–61, Oakland, California, May 1992. IEEE CS.

[21] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, IETF, May 1992.

[22] V. Jacobson, C. Leres, and S. McCanne. libpcap, March 2004. `http://www.tcpdump.org/`.

[23] P. A. Karger and J. C. Wray. Storage channels in disk arm optimization. In *1991 IEEE Symposium on Security and Privacy*, pages 52–63, Oakland, California, May 1991. IEEE CS.

[24] T. Kohno, A. Broido, and k. claffy. Remote physical device fingerprinting. In *2005 IEEE Symposium on Security and Privacy*, pages 211–225, Oakland, California, May 2005. IEEE CS.

[25] M. G. Kuhn. Personal communication.

[26] B. W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[27] M. Martinec. Temperature dependency of a quartz oscillator. `http://www.ijs.si/time/#temp-dependency`.

[28] D. L. Mills. Network time protocol (version 3) specification, implementation and analysis. RFC 1305, IETF, March 1992.

[29] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. Technical Report 98–43, Department of Computer Science University of Massachusetts at Amherst, October 1998.

[30] I. S. Moskowitz, R. E. Newman, D. P. Crepeau, and A. R. Miller. Covert channels and anonymizing networks. In P. Samarati and P. F. Syverson, editors, *Workshop on Privacy in the Electronic Society*, pages 79–88, Washington, DC, USA, October 2003. ACM Press.

[31] I. S. Moskowitz, R. E. Newman, and P. F. Syverson. Quasi-anonymous channels. In M. Hamza, editor, *IASTED Communication, Network, and Information Security*, pages 126–131, New York, USA, December 2003. ACTAPress.

[32] J. A. Muir and P. C. van Oorschot. Internet geolocation and evasion. Technical Report TR-06-05, Carleton University – School of Computer Science, April 2006.

[33] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[34] S. J. Murdoch and S. Lewis. Embedding covert channels into TCP/IP. In M. Barni, J. Herrera-Joancomartí, S. Katzenbeisser, and F. Pérez-González, editors, *Information Hiding: 7th International Workshop*, volume 3727 of *LNCS*, pages 247–261, Barcelona, Catalonia (Spain), June 2005. Springer-Verlag.

[35] R. M. Needham. Denial of service. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 151–153, New York, NY, USA, 1993. ACM Press.

[36] R. M. Needham. Denial of service: an example. *Commun. ACM*, 37(11):42–46, 1994.

[37] R. E. Newman, V. R. Nalla, and I. S. Moskowitz. Anonymity and covert channels in simple timed mix-firewalls. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*. Springer-Verlag, May 2004.

[38] L. Øverlier and P. F. Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2006. IEEE CS.

[39] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In W. Effelsberg, H. W. Meuer, and G. Müller, editors, *GI/ITG Conference on Communication in Distributed Systems*, volume 267 of *Informatik-Fachberichte*, pages 451–463. Springer-Verlag, February 1991.

[40] J. Postel. Internet control message protocol. RFC 792, IETF, September 1981.

[41] R. Redelmeier. CPUBurn, June 2001. `http://pages.sbcglobal.net/redelm/`.

[42] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[43] Reporters Without Borders. Blogger and documentary filmmaker held for the past month, March 2006. `http://www.rsf.org/article.php3?id_article=16810`.

[44] G. Uchenick. MILS middleware for secure distributed systems. *RTC magazine*, 15, June 2006 2006. `http://www.rtcmagazine.com/home/article.php?id=100685`.