Estimating the Capacity of Temperature-based Covert Channels

Sebastian Zander, Philip Branch, Grenville Armitage
Centre for Advanced Internet Architectures, Technical Report 100726A
Swinburne University of Technology
Melbourne, Australia
szander@swin.edu.au, pbranch@swin.edu.au, garmitage@swin.edu.au

***Abstract*—Covert channels aim to hide the existence of communication. Recently, Murdoch proposed a temperature-based covert channel where information is transmitted by remotely inducing and measuring changes of temperature of an unwitting intermediate host. The channel was invented for the purpose of attacking anonymous servers, but could also be used for general-purpose covert communications. We propose an empirical method for estimating realistic (and previously unknown) capacities for this channel. In example scenarios with different intermediate hosts and different levels of temperature induction and noise we find the channel capacity is up to 20.5 bits per hour, but it almost halves to 10.3 bits per hour with higher noise or more effective cooling at the intermediate host.**

***Index Terms*—Security, Temperature-based Covert Channels, Capacity**

I. Introduction

Often the simple fact that communication exists is enough to cause third parties to become suspicious and take further actions. Covert channels aim to hide the very existence of communications [1]. Individuals and groups have various reasons to utilise covert channels, often motivated by the existence of an adversarial relationship between two parties. Examples include government agencies versus criminal or terrorist organisations, hackers or corporate spies versus company IT departments, or dissenting citizens versus their governments.

Many network protocol-based covert channels have been proposed ranging from very simple channels (e.g. encoding covert information in unused header bits) to more complex channels (e.g. encoding covert bits in packet timing) [1]. Recently, Murdoch proposed a temperature-based covert channel that transmits information by measuring remotely-induced changes of temperature of an unwitting intermediate host [2]. Initially proposed to identify anonymous servers, such as hidden services in the Tor anonymisation network [3], Murdoch's channel can also be used for general-purpose covert communications.

In a temperature-based covert channel the covert sender (by convention *Alice*) modulates the CPU load of an unwitting intermediate host or the target host in anonymity attacks by varying the rate of requests sent to it based on the covert bits to be sent. The change in CPU load changes the temperature, which in turn changes the skew of the intermediate host's clock. The covert receiver (by convention *Bob*) probes the intermediate host's clock and recovers the covert bits by estimating the clock-skew changes [2].

Two scenarios are possible. In the first scenario the intermediate host is separated from both Bob and Alice by a network (see Figure 1). Alice and Bob could be controlled by the same person (e.g. attacking Tor hidden services) or by different persons (e.g. general covert communication).

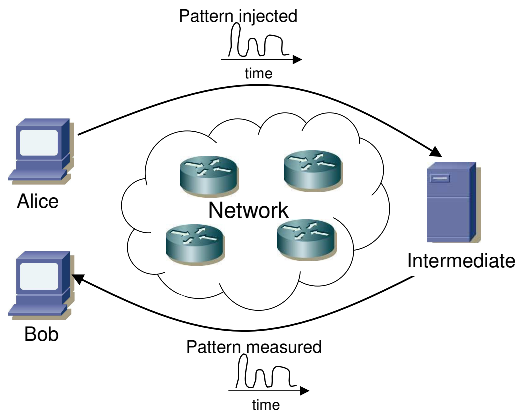


Figure 1. Temperature-based covert channel where Alice and Bob are separated from the intermediate host by a network

In the second scenario Alice is located on the intermediate host and manipulates the CPU load directly. Only Bob is separated from the intermediate host by a network. This is a possible scenario for the ex-filtration

This technical report is a revised version of CAIA-TR-091218A. We improved the clarity of the text in several places and added an analysis of the channel capacity when load is induced remotely through web page requests.

of sensitive information. The usefulness or threat of the channel (according to one's perspective) depends on the channel capacity, which is the maximum transmission rate at which error-free communication is possible [4].

We propose a method to estimate the capacity of Murdoch's temperature-based covert channel. We model the channel as Additive White Gaussian Noise (AWGN) channel [5]. The channel capacity depends on the channel's bandwidth and signal to noise ratio (SNR). Based on empirical measurements we create a simulation model that models the relationship between CPU load and clock skew and use it to estimate the bandwidth and signal power. We estimate the noise power from empirical measurements of the channel without an input signal.

We analyse the capacity in some example scenarios depending on different levels of load inducement and channel noise and different intermediate hosts. With an intermediate host similar to the one in [2] and low noise the capacity is 20.5 bits per hour (bph), but it almost halves to 10.3 bph with higher noise or with similar noise but different intermediate host with more effective cooling.

The paper is organised as follows. In Section II we describe our methodology for estimating the capacity. In Section III we describe the creation of the simulation model including the empirical measurements. In Section IV we examine the channel noise. In Section V we describe how the channel bandwidth is estimated, and present the results. Section VI discusses possible countermeasures against the channel. Section VII concludes and outlines future work.

II. Methodology

Ambient temperature and humidity vary over time affecting the measured clock skew [2]. However, ambient changes usually happen on longer timescales and it is possible to remove these long-term trends from the clock skew output. With this our system is time-invariant as the output depends only on the input and additive noise. The channel suffers from multiple independent sources of noise, and our empirical measurements confirm this is approximately Gaussian.

Thus we model the channel as an Additive White Gaussian Noise (AWGN) channel [5]. The channel capacity is:

$$C = B \cdot \log_2\left(1 + \frac{P}{N}\right) , \quad (1)$$

where B is the bandwidth of the channel and P/N is the signal-to-noise ratio (SNR), the average signal power divided by the average noise power. Figure 2 shows our overall system model. The input of the AWGN channel is a clock skew signal generated by Alice $s_S(t)$. The output of the channel $r(t)$ is the clock skew signal measured by Bob plus the noise $n(t)$.

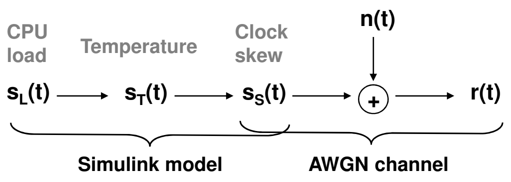


Figure 2. Model of the temperature-based covert channel

But Alice can only indirectly manipulate $s_S(t)$ by modulating CPU load $s_L(t)$. To model the relationship between $s_L(t)$ and $s_S(t)$ through changes of temperature $s_T(t)$ we created a Matlab Simulink [6] model (see Section III). The model allows simulating a frequency sweep to estimate the channel's bandwidth. This reduces the number of testbed experiments drastically, as only a few experiments are needed to calibrate the model. We also use the model to estimate the signal power. We investigate the effects of different CPU load inducement and different intermediate hosts on the bandwidth and signal power (see Section V).

The noise $n(t)$ includes noise introduced by the clock skew estimation (network jitter and timestamp quantisation noise [2], [7]) as well as noise because of CPU load or temperature fluctuations. We estimate the noise power for a particular intermediate host from empirical measurements of $r(t)$ without any input signal $s_S(t)$ (see Section IV). We measure the noise during day and night and for different amounts of background CPU load (load not caused by Alice).

Timestamp quantisation noise was minimised by using the improved clock skew estimation technique from [7]. As in [2] we assume relatively ideal network and ambient conditions. Our testbed was small and only lightly utilised and hence network jitter was small. Ambient temperature/humidity changes were also small, as all PCs were located in air-conditioned rooms. Therefore, our results represent upper bounds for scenarios with "worse" conditions. However, even on long uncongested paths through the Internet jitter is typically skewed towards low values [7] and usable intermediate hosts (servers) are often in air-conditioned rooms.

III. Simulink Model

We first describe the experiments carried out for fitting the model and then describe the model.

A. Experiments

The first intermediate host (IH1) was a 2.4 GHz Intel Celeron CPU inside a midi-tower case running Linux

2.6. Both CPU and power supply fans ran at constant speed. The second intermediate host (IH2) was a 2.8 GHz Intel Pentium CPU inside a desktop case running FreeBSD 4.10. It had a more effective thermally-controlled CPU fan designed so that most of the warm air is directly blown out of the case. Alice introduced CPU load locally on IH1 or IH2 using the load generator cpuburn [8] or remotely from another PC by repeatedly fetching a small static web page from an Apache 2.2.3 web server [9] with Secure Socket Layer (SSL) enabled. Bob was another PC. All PCs were connected to the same network switch.

We conducted three sets of experiments:

1) Alice generated load on IH1 locally with cpuburn.
2) Alice generated load on IH1 remotely via requesting web pages.
3) Alice generated load on IH2 locally with cpuburn.

In our experiments we generated periodic square wave signals $s_{\mathrm{L}}(t)$ and remotely measured the resulting $s_{\mathrm{S}}(t)$. Each signal period of the periodic signal was a time of maximum induced CPU load (approximately 100% load) followed by the same time of idle CPU (approximately 0% load) allowing the system to cool down to its previously unloaded temperature. Each experiment consisted of 10 consecutive signal periods. We ran separate experiments using load-inducement times of 180, 300, 600, 1200, 1800, 2400 and 3600 seconds. The induced CPU load was ~100% with cpuburn, but with ~30 web requests per second it was only ~69% (similar to experiments in [2]).

Changes in the ambient temperature or humidity affect the clock crystal and hence introduce noise. During our experiments we tried to minimise this noise. PCs were located in air-conditioned rooms and we performed measurements during times when doors and windows were closed and no humans were inside the rooms. Nevertheless, there were some changes in the ambient conditions. However, these ambient changes usually happen on longer timescales and it is possible to remove long-term trends from the measured $s_{\mathrm{S}}(t)$. Furthermore, if the induced clock-skew change is large the noise has a much smaller magnitude than the signal.

Clock skew can only be estimated rather than directly measured with noise being introduced by network jitter and timestamp quantisation [2]. We queried the TCP clock [10] of the intermediate hosts with an average frequency of 1 Hz. Since we used the improved clock-skew estimation technique [7], the timestamp quantisation noise was very low despite low clock frequencies of 250 Hz (IH1) and 100 Hz (IH2). Connecting Alice and Bob to the same switch minimised network jitter. But even for long uncongested paths network jitter is often skewed towards low values [7].

One clock-skew estimate is obtained for time windows of w seconds, containing w clock samples in our case. If w is set too small, the clock skew estimates contain a lot of noise. On the other hand too large w lead to averaging which prevents the accurate measurement of steep changes. In order to get more frequent clock-skew estimates without reducing the window size oversampling can be used [7]. We examined window sizes of 60, 120, 180, 240, 300 and 600 seconds. We found that for 60 s and 120 s windows the noise was very high whereas for windows of 240 s and larger there is too much averaging.

Therefore, for IH1 we selected $w = 180$ s. However, for IH2 the induced clock-skew changes are almost one magnitude smaller. To limit the noise to acceptable levels we used $w = 600$ s. Oversampling was used to get one clock skew estimate every 30 s regardless of w. Note that consistent with [2] throughout the paper we always use *negated* clock skew since usually negated clock skew is proportional to temperature and CPU load.

B. *Simulation Model*

We developed the simulation model using Matlab Simulink [6]. The input of the model is a CPU load signal $s_{\mathrm{L}}(t)$ (values ranging from 0 to 1) and the output of the model is the clock-skew signal $s_{\mathrm{S}}(t)$ (values in parts per million). The model is loosely based on Simulink's thermal model of a house, but is more complex since it has two heat capacitors instead of one. One capacitor has a larger capacity and heats up slower while the other has a smaller capacity and heats up quicker. We hypothesise that the first capacitor is the inside of the PC's case, while the second capacitor is the CPU and heat sink.

For each set of experiments with different load inducement we fitted the model based on the empirical data. Figure 3 shows the model for IH1 with remote load inducement (the integrators' capacities are $C_1 = 2.625^{-6}$ and $C_2 = 1.625^{-6}$). However, the structure of the model is generic. For the other cases we use the same model but with different gain constants and heat capacities. The model should be applicable to other intermediate hosts, as the overall shape of CPU load induced clock-skew changes looks similar for other PCs [2].

We used the same CPU load signals previously used in the empirical measurements as input for the model. Figure 4 compares the clock-skew change experimentally measured (average over all 10 signal periods) with the model output. Overall there is a very good match. For the shortest load inducement times the measured peaks are slightly lower than the predicted peaks because of

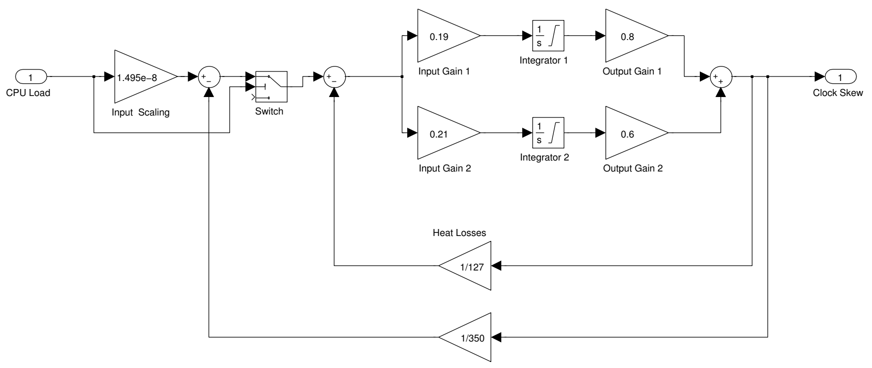


Figure 3. Matlab Simulink model for simulating the relationship between CPU load and clock skew (parametrised for intermediate host 1 with remote load inducement via web requests)

the averaging effect. Despite the larger window size the empirical data for IH2 is much noisier since the clock-skew changes are significantly smaller. Note that all clock skew estimates have been detrended from long-term ambient temperature changes and normalised to allow direct comparison. Hence Figure 4 shows relative changes of clock skew rather than absolute values.

With ambient changes removed our experimental system is time-invariant as $s_{\mathrm{S}}(t)$ depends only on $s_{\mathrm{L}}(t)$. Ideally our system would also be linear, as linear time invariant system are easy to analyse. There are several dependencies that are non-linear in general, but within our operating conditions we assume them to be roughly linear. In general temperature does not change linearly with CPU load, but depends on the mix of instructions executed as well as possible CPU frequency changes. Since our CPUs ran with constant frequency and we always generated load with cpuburn we assume this relation is roughly linear. In general clock-skew does not change linearly with CPU load, but for typical temperatures inside PC cases the relation is also roughly linear [2].

Unfortunately, our system shows non-linear behaviour. The thermally-controlled CPU fan in IH2 results in a very quick settling of the temperature compared to a constant-speed fan. For both intermediates the cooling down is slower than the heating up because loading the CPU introduces additional energy, but when the CPU is idle there is no additional energy introduced for cooling (the thermally controlled fan returns to its lowest speed immediately after the load inducement stops).

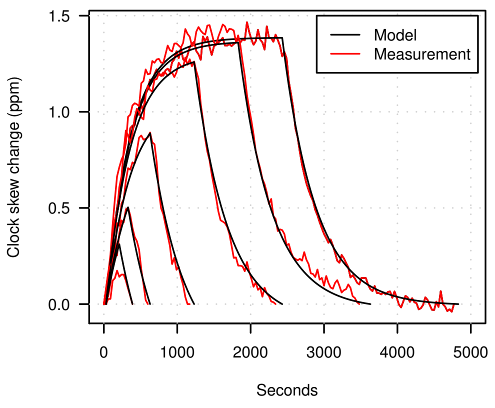
We generated linear models from the non-linear models using the Matlab Simulink model linearisation function *linmod* (linearisation for individual blocks based on pre-programmed analytic block Jacobians). For IH1 the linear model matches quite well, although it does deviate slightly in the cool-down phase. For IH2 the linear model does not match well because it cannot capture the very steep temperature rise and settling. Therefore, we use simulation with the non-linear models instead of analysing the linear models to determine the bandwidth of the channel (see Section V).

IV. Channel Noise

In order to estimate the noise on the channel we measured clock-skew changes of the intermediate hosts. For IH1 and IH2 we measured the clock-skew changes without any CPU load inducement (idle machines at ~0% CPU load). For IH1 we also measured the clock-skew changes with load on the web server (referred to as *background load*). We measured the noise when IH1 was lightly loaded with ~1 web request per second (~3% CPU load) and more heavily loaded with ~10 web requests per second (~25% CPU load). In some experiments we also measured the temperature inside the room and the PC case.

First we show the temperatures (both normalised on the minimum temperature of each series) and the remotely measured variable clock skew for an idle IH1. Figure 5 shows a few hours in the afternoon/evening. The case temperature is fluctuating within a few 0.1 degrees Celsius without a clear trend and does not closely follow the room temperature before 20:00 hours. Overall the variable clock skew looks similar to random noise. Figure 6 also shows 8–9 hours during the night when the room and case temperature are decreasing and the clock skew is decreasing accordingly. Thus the variable clock skew is not random but has a clear trend following the trend of the ambient temperature.

During the day temperature changes inside the case are not highly correlated with the room-temperature changes. This is probably because IH1 was located in close proximity to two other PCs that were actively used during the day. Another factor introducing noise during the day could have been the presence of a human – also in close proximity to IH1. During the night when all



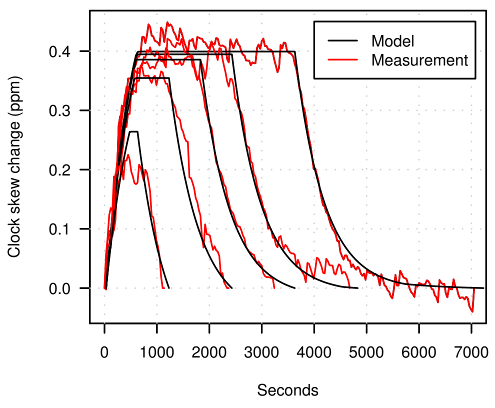


Figure 4. Comparison of normalised clock-skew output of simulation model and empirical measurements for intermediate host 1 with remote load inducement (left) and intermediate host 2 with cpuburn load inducement (right)

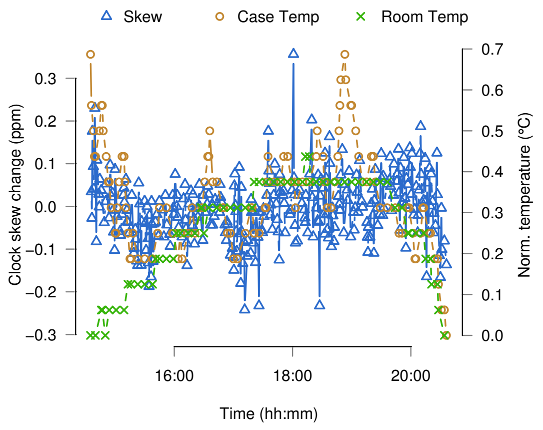


Figure 5. Variable clock skew and case/room temperature during afternoon/evening (intermediate host 1)

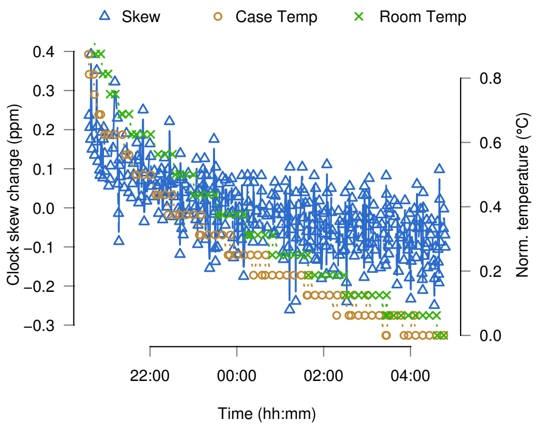


Figure 6. Variable clock skew and case/room temperature during night (intermediate host 1)

PCs were idle and no humans were present, the case temperature shows exactly the same change over time as the room temperature (but at 3.2 degrees Celsius higher). For IH2 the results are similar.

The noise is clearly not Gaussian because of the ambient trends, but we can detrend the data. We used a LOWESS smoother [11] to compute a smoothed series of data points. The detrended series was then computed by subtracting the smoothed series from the actual data series. In the following we investigate whether the detrended noise has a Gaussian distribution.

We used the Shapiro-Wilk statistical test of normality. Table I in the Appendix shows the test statistics and p-values for the different noise signals (1% outliers removed at each edge). For an idle IH1 we cannot reject the hypothesis that the data is Normally distributed (99% significance level). However, in the other cases we cannot draw the same conclusion, as the resulting p-values are too low. However, the Shapiro-Wilk test is very sensitive to small deviations.

Figure 7, Figure 8 and figure 9 show quantile-quantile (QQ) plots of the empirical distributions against the theoretical Gaussian distribution for IH1 (with and without background load) and IH2. In all graphs the points follow the quantile-quantile line closely, except at the edges. This indicates that the empirical distributions are roughly Gaussian, except for some outliers[1].

V. Channel Capacity

The temperature-based covert channel is basically a base-band system acting as a low-pass filter on the input signal. To estimate the bandwidth we need to estimate the upper cut-off frequency, which is commonly defined as the frequency where the power of the output has decreased by 3 decibel (dB). We estimated the bandwidth by simulating different signal period lengths with the

[1]Overall, day noise fits a Gaussian distribution better than night noise, which may indicate that our detrending algorithm could be further improved.

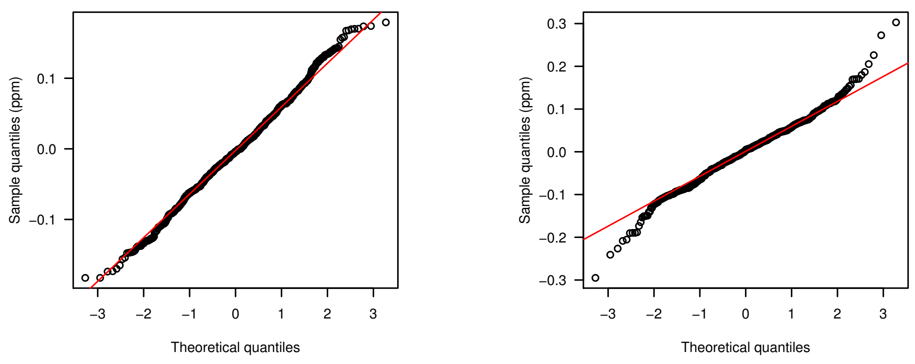


Figure 7. QQ plots of detrended variable clock skew when idle during day (left) and night (right) for intermediate host 1 vs. Gaussian distribution

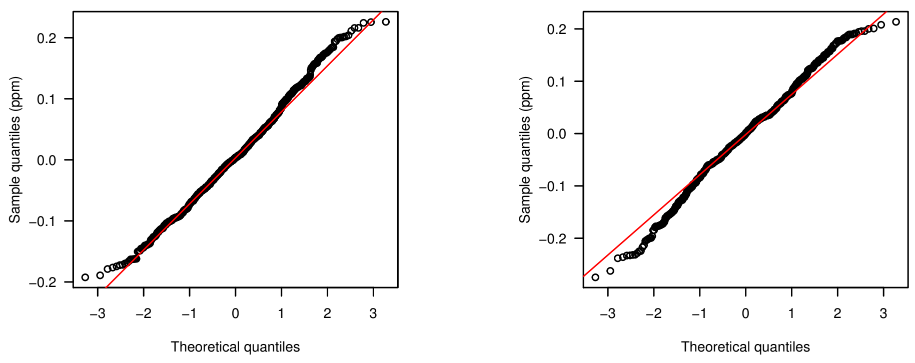


Figure 8. QQ plots of detrended variable clock skew with background load caused by ~1 web page request/second (left) and ~10 web page request/second (right) for intermediate host 1 vs. Gaussian distribution

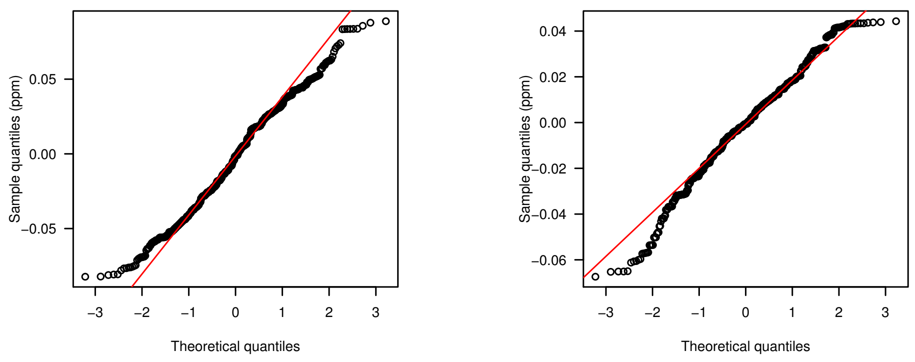


Figure 9. QQ plots of detrended variable clock skew when idle during day (left) and night (right) for intermediate host 2 vs. Gaussian distribution

Simulink model and identifying when the power has decreased by approximately 3 dB.

The channel bandwidths are $B_1 \approx 0.000451\,\text{Hz}$ for IH1 with cpuburn, $B_2 \approx 0.000422\,\text{Hz}$ for IH1 with web request load and $B_3 \approx 0.000444\,\text{Hz}$ for IH2 with cpuburn. The results are consistent with the results from the empirical measurements (see Figure 4). For example, B_2 equals square impulses with 1185 s load inducement (half the period) and the measured 1200 s load inducement signal has approximately 51% of the power, which is exactly what the model predicts.

Figure 10 shows the channel capacity based on the SNR in dB for IH1 with cpuburn (C_1), IH1 with web requests load (C_2) and IH2 with cpuburn (C_3). The capacity increases almost linearly with the SNR for larger SNRs. However, the sending power is not unlimited and hence the capacity cannot increase to infinity. The question is: what SNRs can be achieved?

We estimated the average power of signal and noise by computing the power spectral density (power per frequency band), integrating over all frequency bands within the channel bandwidth and then normalising the

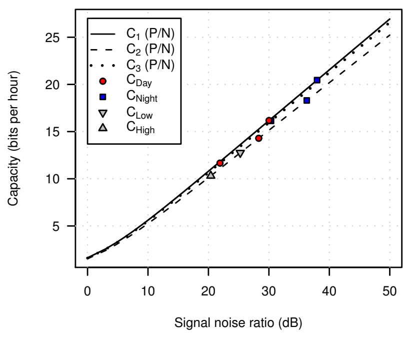


Figure 10. Channel capacity based on signal-to-noise ratio for all three scenarios; the points C_{Day} and C_{Night} depict the capacities given the empirical noise during the day and night for each capacity curve and the points C_{Low} and C_{High} depict the capacities given the empirical noise caused by low rate and high rate web traffic during the day

power based on the number of samples of the signal. We computed the average signal power based on the model output for alternating 100% and 0% CPU load with a frequency equal to the channel bandwidth. We computed the average noise power from the empirically measured detrended noise signals. Figure 10 shows the capacities C_{Day} and C_{Night} for the SNRs during day and night on all curves and C_{Low} and C_{High} for the SNRs with noise caused by low and high rate web requests.

As expected, the SNR is higher during the night and the SNR decreases with increasing background load of the web server. The capacity is about 20.5 bph for IH1 with maximum signal power (cpuburn) and minimum noise (idle during night). Remote load inducement with minimum noise (idle during night) reduces that capacity to only 18.3 bph, but with increasing background server load it reduces to 10.3 bph. IH2 with much more effective cooling decreases the capacity significantly to 16.1 bph (idle during night).

Overall, our capacity estimates are significantly higher than the very rough estimate of 2–8 bph [2]. However, the initial estimate was ad-hoc based on an inspection of the experimental data and could not consider the improved clock-skew measurement technique developed later [7].

The capacity of the channel is more than sufficient for attacking anonymous servers since in this scenario only a few bits need to be transmitted. The probability of choosing a wrong host from the candidate set (a false positive) is $p_{\mathrm{FP}} = 2^{-n}$ where n is the number of covert bits transmitted [2]. For example, even if only 16 bits are transmitted the probability of a false positive is only $p_{\mathrm{FP}} = 1.525879^{-5}$. Given our capacity estimates it takes only 1–2 hours to transmit 16 bits. However, for general-purpose communications the capacity is very small and there exist other network covert channels with higher capacities [1]. On the other hand there may be some situations in which this is the only available channel [2].

The existence of covert channels with capacities of less than one 1 bit/s is deemed acceptable in many application environments [12]. However, in scenarios where only a few transmitted bits pose a security threat, such as in anonymisation networks, temperature-based covert channels require handling. We discuss possible countermeasures in the following section.

VI. Countermeasures

It is very difficult to completely eliminate the temperature-based covert channel. However, a number of measures could be employed to reduce its capacity.

A seemingly obvious way of eliminating the channel is to prevent Bob's remote sampling of the clock by removing all timestamps from all network protocols, which could be done by the intermediate host or a security gateway. However, removing the timestamps negatively affects the performance and functionality of protocols. For example, the TCP timestamp extension is needed for improving performance of TCP and the HTTP timestamp is needed for HTTP caching. Furthermore, many low-level operating system events are triggered on timer interrupts and could be remotely detected and used instead of explicit timestamps [10].

A clock crystal that is not affected by temperature changes eliminates the channel. However, temperature-compensated clock crystals might not have adequate accuracy [13]. Oven-compensated crystals have good accuracy, but are very expensive and power hungry [13]. Thus it seems unlikely that accurately compensated crystals would ever be deployed on a large scale.

The opportunity for Alice to induce CPU load cannot be completely eliminated. However remote load inducement could be limited if the network traffic is throttled before it reaches the intermediate host or in the worst case on the intermediate host itself before it reaches an application. If Alice is located on the intermediate a similar measure is to limit the amount of CPU time a process or user can use.

Another countermeasure is to increase the channel noise by randomly varying CPU load on the intermediate host or in the extreme case by continuously running the CPU at full load. However, this strategy is obviously very inefficient. Furthermore, care must be taken in the implementation because the temperature does not only depend on the CPU load but also on the specific mix of instructions executed and hence different types of tasks can have different temperature effects [13].

Detection of the covert channel is also not straightforward. A detector has to look for abnormal traffic or

CPU load patterns indicating either Alice or Bob. Hence the detection accuracy depends on how normal patterns look like at the intermediate host. Furthermore, Alice and Bob could always vary their traffic or CPU load patterns within some limits trying to evade detection, although this would most likely also reduce the throughput.

VII. Conclusions and Future Work

We proposed a method for estimating the capacity of Murdoch's temperature-based covert channel [2]. We developed a Simulink simulation model for modelling the relation between CPU load and clock skew and used it to determine the channel's bandwidth and signal power. We estimated the noise power based on empirical measurements and also showed that the detrended noise is roughly Gaussian. Therefore, we estimated the capacity using the well-known AWGN channel model. With an intermediate host similar to the one in [2] and low noise the capacity is 20.5 bits per hour (bph), but it almost halves to 10.3 bph with higher noise or with an intermediate host with more effective cooling.

The capacity is more than sufficient for attacking anonymous servers, but for general-purpose communications it is quite small. Furthermore, in many scenarios there are other network covert channels with higher capacities available [1]. On the other hand in some situations the temperature-based covert channel may be the only usable channel [2].

In future work we aim to examine a larger number of different PCs and compare their channel capacities. We also plan to use our Simulink model in combination with the Matlab Simulink communications toolbox [6] to measure the throughput of the channel depending on different encoding techniques.

Acknowledgements

We thank Steven Murdoch, Markus Kuhn and Andrew Moore for their insightful comments and support during the initial part of this research carried out while Sebastian Zander was visiting the Computer Lab of Cambridge University.

References

[1] S. Zander, G. Armitage, P. Branch, "A Survey of Covert Channels and Countermeasures in Computer Network Protocols," *IEEE Communications Surveys and Tutorials*, vol. 9, pp. 44–57, October 2007.

[2] S. J. Murdoch, "Hot or Not: Revealing Hidden Services by Their Clock Skew," in *Proceedings of 13th ACM Conference on Computer and Communications Security (CCS)*, pp. 27–36, November 2006.

[3] R. Dingledine, N. Mathewson, P. Syverson, "Tor: The Second-generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[4] C. E. Shannon, "A Mathematical Theory of Communications," *Bell Systems Technical Journal*, vol. 27, July 1948.

[5] T. M. Cover, J. A. Thomas, *Elements of Information Theory*. Wiley Series in Telecommunications, John Wiley & Sons, 1991.

[6] The Mathworks, "Simulink – Simulation and Model-Based Design." http://www.mathworks.com/products/simulink/.

[7] S. Zander, S. J. Murdoch, "An Improved Clock-skew Measurement Technique for Revealing Hidden Services," in *Proceedings of Usenix Security*, July/August 2008.

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

[9] Apache Software Foundation, "Apache Web Server." http://www.apache.org/.

[10] T. Kohno, A. Broido, kc claffy, "Remote Physical Device Fingerprinting," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 211–225, May 2005.

[11] W. S. Cleveland, "LOWESS: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression," *The American Statistician*, vol. 35, no. 1, 1981.

[12] US Department of Defense Standard, "Trusted Computer System Evaluation Criteria," Tech. Rep. DOD 5200.28-STD, US Department of Defense, December 1985. http://csrc.nist.gov/publications/history/dod85.pdf.

[13] S. J. Murdoch, S. Zander, "Hot or Not: Fingerprinting Hosts Through Clock Skew," in *Presentation at EuroBSDCon*, September 2007. http://www.cl.cam.ac.uk/~sjm217/talks/eurobsdcon07hotornot.pdf.

Appendix

Table I shows the test statistic values and p-values for the Shapiro-Wilk normality test performed for the different clock-skew datasets (see Section IV).

Table I
Shapiro-Wilk test statistics and p-values

Intermediate	**Noise**	**Statistic (W)**	**p-value**
IH1	**Day**	0.997	4.6%
IH1	**Night**	0.996	1.9%
IH1	**Light web load**	0.994	0.08%
IH1	**Moderate web load**	0.994	0.08%
IH2	**Day**	0.992	0.05%
IH2	**Night**	0.91	$\ll 1\%$