Estimating the Capacity of Temperature-based Covert Channels

Sebastian Zander, Philip Branch, Grenville Armitage
Centre for Advanced Internet Architectures, Technical Report 091218A
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 between two or more parties. Such channels typically utilise pre-existing overt data transmissions to carry hidden messages. Recently, Murdoch proposed a temperature-based covert channel where information is transmitted by remotely inducing and measuring changes of temperature of an intermediate/target host. The channel was invented mainly for the purpose of attacking anonymous servers, but could also be used for general-purpose covert communications. We propose a method for estimating the channel capacity, which largely depends on the intermediate host. Evaluation of two different intermediate hosts shows upper bounds for the channel capacity of 10.8–15.4 and 16.4–21.1 bits per hour depending on the noise.**

***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 where covert information is transmitted by remotely inducing and measuring changes of temperature [2]. Initially proposed for identifying 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).

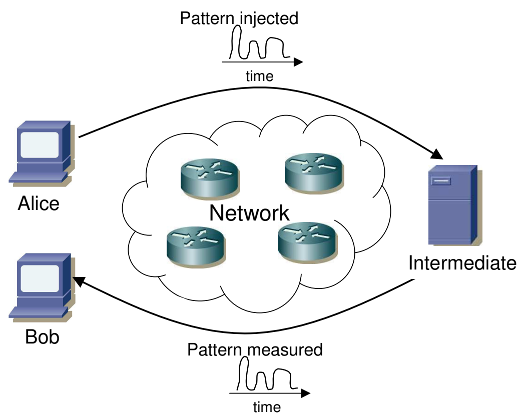


Figure 1. Temperature-based covert channel where Alice and Bob are separated from the intermediate host by a network

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 of sensitive information. To determine the usefulness or threat (depending on the viewpoint) of the channel it is necessary to know 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). 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. Our empirical evaluation of two different intermediate hosts shows that the channel capacity is 10.8–15.4 and 16.4–21.1 bits per hour depending on the noise. Since we assumed ideal conditions for Alice and Bob these capacities are upper bounds.

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 and affect 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 some additive noise. Since the channel suffers from multiple independent sources of noise we assume the noise is approximately Gaussian (Central Limit Theorem), which is confirmed by our empirical measurements.

Therefore, we model the temperature-based channel as AWGN channel and the channel capacity can be computed based on the channel bandwidth and SNR [5]. Figure 2 shows our channel 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. The noise $n(t)$ includes noise introduced by the clock skew estimation (mainly network jitter and timestamp quantisation noise [2], [6]) as well as noise caused by 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 $s_S(t)$. With ambient temperature trends the noise is not Gaussian, but we show that de-trended

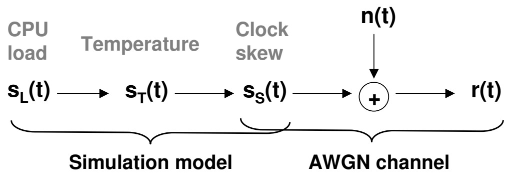


Figure 2. Model of the temperature-based covert channel

noise approximately follows a Gaussian distribution (see Section IV).

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 create a simulation model (see Section III). The model needs to be parametrised according to the empirically measured behaviour of a particular intermediate host (see Section III). Using this simulation model we estimate the channel bandwidth and the signal power for particular intermediate hosts (see Section V).

In our evaluation we assume ideal conditions for Alice and Bob. Alice's signal power is maximal because Alice is located on the intermediate host and CPU load is controlled directly (scenario 2). On the other hand we assume the noise is minimal. The improved clock skew estimation technique [6] is used and hence timestamp quantisation noise is minimised. Network jitter is minimal as our testbed is small. Ambient temperature changes are also small because all PCs were located in climate-controlled rooms.

III. Creating the Simulation Model

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

A. Experiments

The first intermediate host (intermediate 1) 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 (intermediate 2) 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 was located on the intermediate host. Bob was a second PC running Linux 2.6 connected to the same network switch.

CPU load was induced using cpuburn [7]. During the experiments no network traffic, besides Bob's clock probes, was send to the intermediate host. Also, no other processes ran that used significant amounts of CPU time. However, housekeeping processes used minimal CPU resources and could have caused some network traffic.

In our experiments we generated periodic square wave signals $s_L(t)$ and remotely measured the resulting $s_S(t)$. 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.

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 climate-controlled rooms and we performed measurements during times where 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 $s_S(t)$. 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 [8] of the intermediate hosts with an average frequency of 1 Hz. Since we used the improved clock-skew estimation technique [6], the timestamp quantisation noise was very low despite low clock frequencies of 250 Hz (intermediate 1) and 100 Hz (intermediate 2). 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 [6].

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 [6]. Figure 3 shows the estimated clock skew for 3600 s load inducement on intermediate 1 for $w = 120$ s and $w = 600$ s (average over the 10 periods). The oversampling factor was chosen so that one clock-skew estimate is obtained 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.

The figure clearly shows the higher noise with 120 s windows. On the other hand the averaging caused by 600 s windows is also clearly visible when looking at the steep clock-skew decrease occurring when the load

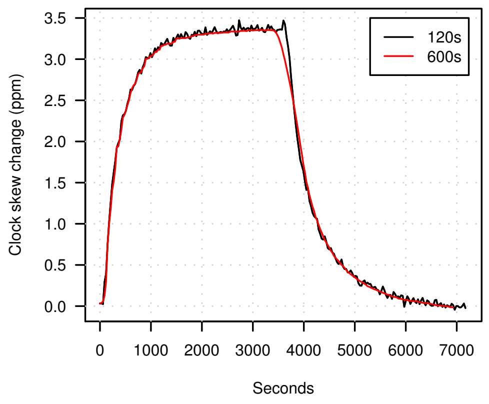


Figure 3. Effect of clock-skew estimation window size on the shape of the received signal

inducement ends. This averaging is more problematic for shorter load-inducements. 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 intermediate 1 we selected $w = 180$ s. However, for intermediate 2 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.

B. Simulation Model

We developed the simulation model using Matlab Simulink [9]. The input of the model is a CPU load signal $s_L(t)$ (values ranging from 0 to 1) and the output of the model is the clock-skew signal $s_S(t)$ (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.

Figure 4 shows the model for intermediate 1 (the capacities which are not shown in the figure are $C_1 = 2.625^{-6}$ and $C_2 = 1.675^{-6}$). The gain constants and capacities are specific for intermediate 1 and were fitted based on the empirical data. However, the structure of the model is generic. For intermediate 2 we use the same model but with different gain constants and heat capacities. We believe the model could be applied to other potential intermediate hosts as the overall shape of CPU load induced clock-skew changes looks similar for other PCs [2].

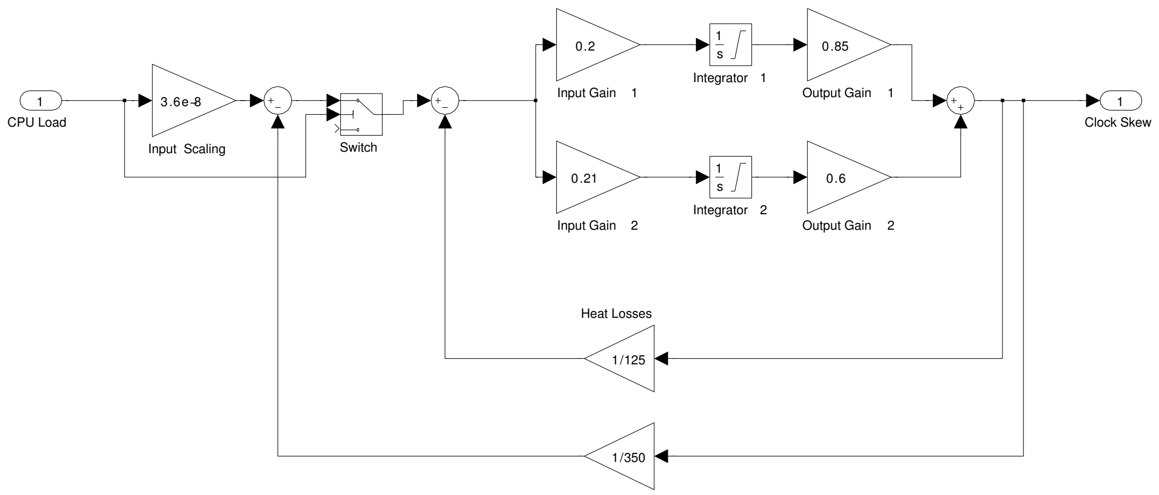


Figure 4. Matlab Simulink model for simulating the relationship between CPU load and clock skew (parametrised for intermediate host 1)

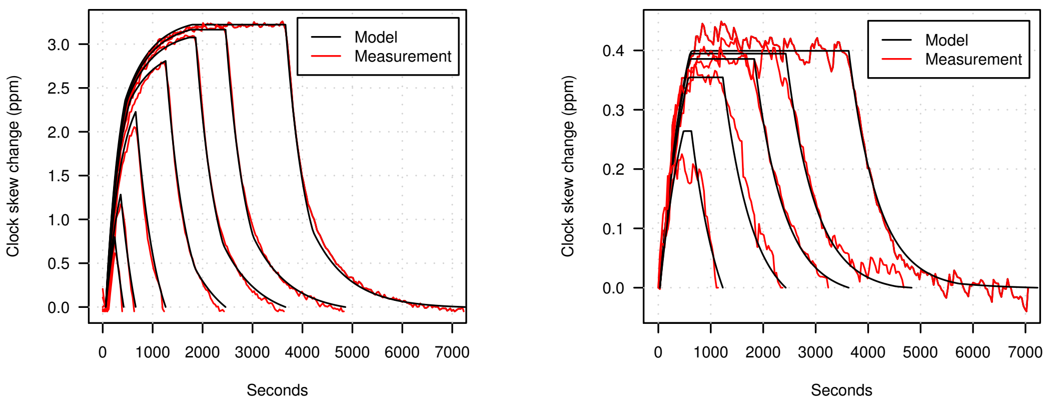


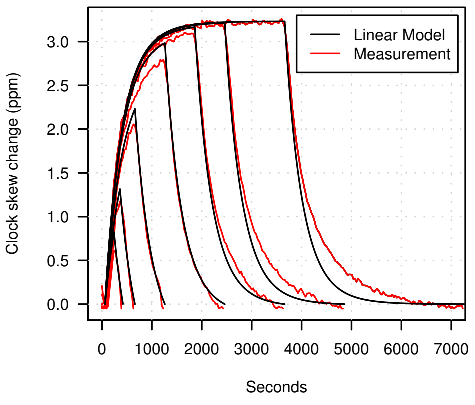
Figure 5. Comparison of normalised clock-skew output of simulation model and empirical measurements for intermediate host 1 (left) and intermediate host 2 (right)

We used the same CPU load signals previously used in the empirical measurements as input for the model. Figure 5 compares the clock-skew change experimentally measured (average over all 10 signal periods) with the output of the model. 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 the averaging effect. Despite the larger window size the empirical data for intermediate 2 is much noisier given that the clock-skew changes are almost one magnitude smaller. Note that all clock skew estimates have been de-trended from long-term ambient temperature changes and normalised to allow direct comparison. Hence Figure 5 shows relative changes of clock skew rather than the real absolute values.

With ambient changes removed our system is basically time-invariant as $s_S(t)$ depends only on $s_L(t)$. Ideally our system would also be linear, as linear time invariant system are easy to analyse. Unfortunately, our experimental system shows non-linear behaviour.

Key sources of non-linearity are the thermally-controlled CPU fan (intermediate 2 only) that 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).

There are other 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].



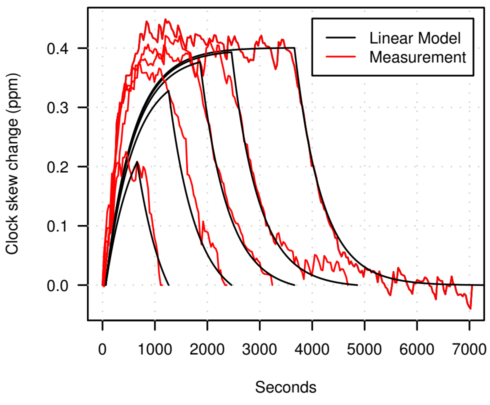


Figure 6. Comparison of normalised clock-skew output of *linearised* simulation model and empirical measurements for intermediate host 1 (left) and intermediate host 2 (right)

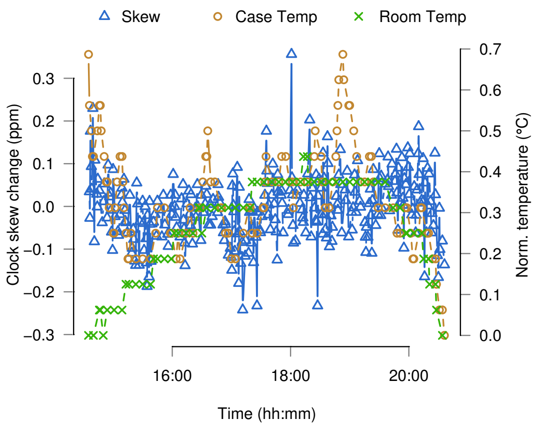


Figure 7. Variable clock skew and case/room temperature during afternoon/evening (intermediate host 1)

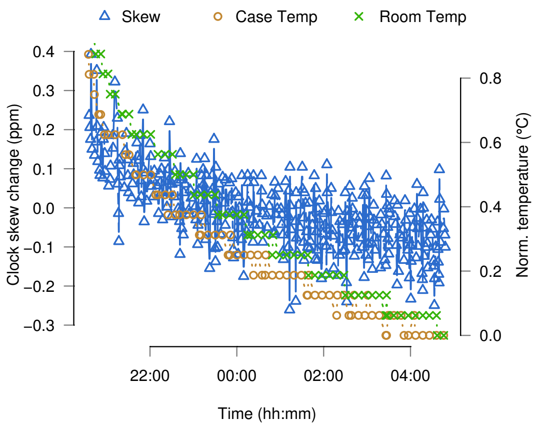


Figure 8. Variable clock skew and case/room temperature during night (intermediate host 1)

For each intermediate host we generated a linear model from the non-linear model using the Matlab Simulink model linearisation function *linmod* (linearisation for individual blocks based on pre-programmed analytic block Jacobians). Figure 6 compares the output of the linearised model with the de-trended normalised empirical data (average over all 10 signal periods). For intermediate 1 the linear model matches quite well, although it does deviate slightly in the cool-down phase. For intermediate 2 the linear model does not match as well because it cannot capture the very steep temperature rise and settling.

IV. Channel Noise

In order to estimate the noise on the channel we measured clock-skew changes of the intermediate hosts, but without any CPU load inducement. We also measured the temperature inside the room and the PC case.

In the following graphs we show the temperatures (both normalised on the minimum temperature of each series) and the remotely measured variable clock skew for intermediate 1. Figure 7 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 8 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 the intermediate PC 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 the intermediate PC. During the night when all PCs were idle and no humans were

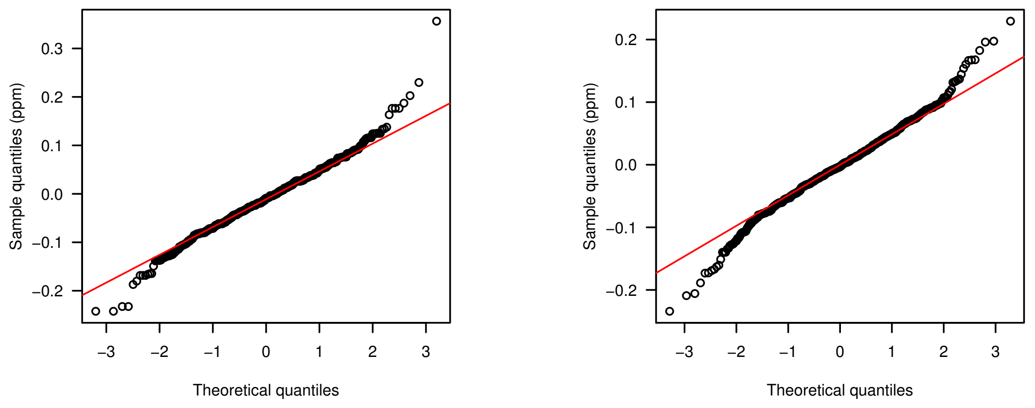


Figure 9. QQ plots of de-trended variable clock skew during day (left) and night (right) for intermediate host 1

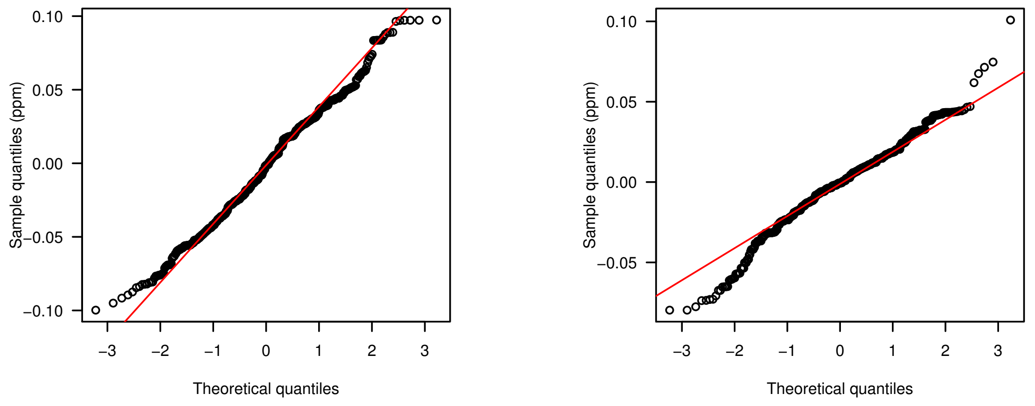


Figure 10. QQ plots of de-trended variable clock skew during day (left) and night (right) for intermediate host 2

present, the case temperature shows exactly the same change over time as the room temperature (but at 3.2 degrees Celsius higher). For intermediate 2 the results are similar.

The noise is clearly not Gaussian because of the ambient changes. However, we de-trended the data from the ambient changes. We used a LOWESS smoother [10] to compute a smoothed series of data points. The de-trended series was then computed by subtracting the smoothed series from the actual data series. In the following we investigate whether the de-trended noise has a Gaussian distribution.

We used the Shapiro-Wilk statistical test of normality. Table I in the Appendix shows the statistics for all data and for 96% of the data (2% outliers removed at each edge). For intermediate 1 with outliers removed we cannot reject the hypothesis that the data is Normally distributed (99% significance level). However, for intermediate 2 we cannot draw the same conclusion, as the resulting p-values are too low.

Figure 9 and figure 10 show quantile-quantile (QQ) plots of the empirical distributions against the theoretical Normal distribution for both intermediate hosts. 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.

V. Channel Capacity

The channel is time-invariant, and we showed that the noise is approximately Gaussian. Therefore, we model the temperature covert channel as an AWGN channel. The capacity of this channel is [5]:

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

where B is the bandwidth of the channel, P is the average signal power and N is the average noise power.

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). For intermediate 1 we compute the bandwidth directly from the linear simulation model (Bode plot). For intermediate 2 the linear model is not a good fit and we estimate the bandwidth by simulating different signal period lengths and identifying when the power has decreased by approximately 3 dB.

The channel bandwidths are $B_1 \approx 0.000434$ Hz for intermediate 1 and $B_2 \approx 0.000455$ Hz for intermediate 2. This means the period of the signal is approximately 2304 s (intermediate 1) and 2250 s (intermediate 2), which is equivalent to 1152 s load followed by 1152 s idle time and 1125 s load followed by 1125 s idle time. This is broadly consistent with the results from the empirical measurements (see Figure 5). For intermediate 1 the measured 1200 s load inducement signal has approximately 52% of the power, which is exactly what the model predicts. For intermediate 2 the measured 1200 s load inducement signal has approximately 47% of the power. The model predicts roughly 53%, but it is hard to measure the low signal power exactly.

Figure 11 shows the channel capacity based on the SNR in dB for intermediate 1 (C_1) and intermediate 2 (C_2). 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 estimate 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 power based on the number of samples of the signal. We compute 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 compute the average noise power from the empirically measured de-trended noise signals, separately for day and night.

As expected for both intermediates the SNRs are higher during the night, as there was more noise during the day. For intermediate 2 the SNRs are smaller despite lower noise because of the much smaller signal power. Figure 11 shows the capacities C_{Day} and C_{Night} for the day and night SNRs on both capacity curves.

Depending on the noise the capacity is between 0.0046 bits/s and 0.0059 bits/s (intermediate 1) and between 0.0030 bits/s and 0.0043 bits/s (intermediate 2). This equates to around 16.4–21.1 and 10.8–15.4 bits per hour. Note that these estimates are upper bounds, because we assumed the noise power to be minimal (idle intermediate host) and the signal power to be maximal (100% CPU load with cpuburn). In reality the capacity is likely to be lower, because the CPU load that the (remote) covert sender can generate is smaller, and the noise on the channel is likely to be larger (higher CPU load jitter and/or higher network jitter).

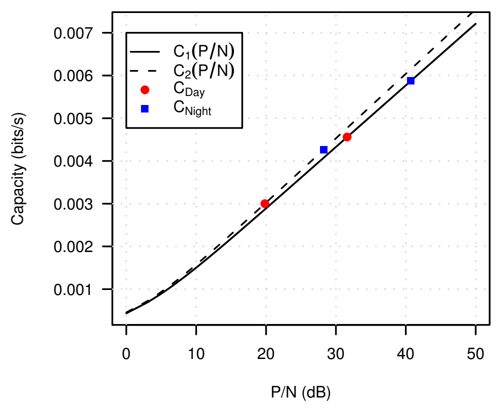


Figure 11. Channel capacity for both intermediate hosts based on signal-to-noise ratio, where the points C_{Day} and C_{Night} depict the capacities given the empirically measured channel noise during the day and night on each capacity curve

Our capacity estimates are significantly higher than the initial estimate of 2–8 bits per hour [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 [6].

The capacity of the channel is 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 $p_{FP} = 2^{-n}$ 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 $p_{FP} = 1.525879^{-5}$. Given our capacity estimates it takes only 1–2 hours to transmit 16 bits.

However, using the channel for general-purpose communications seems impractical, unless very short messages are sufficient. The channel also seems somewhat irrelevant, as there exist many 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 [11]. However, in scenarios where only few transmitted bits pose a threat to security, 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 [8].

A clock crystal that is not affected by temperature changes eliminates the channel. However, temperature-compensated clock crystals might not have adequate accuracy [12]. Oven-compensated crystals have good accuracy, but are very expensive and power hungry [12]. 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 (scenario 1) 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 (scenario 2) 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 [12].

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 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 empirically and also showed that the de-trended noise is roughly Gaussian. Therefore, we estimated the capacity using the well-known AWGN channel model. For two different intermediate hosts we showed that the upper bounds of the channel capacity are 10.8–15.4 and 16.4–21.1 bits per hour respectively.

The capacity is sufficient for attacking anonymous servers, but for general-purpose communications it is quite small. Furthermore, in most 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 simulation model in combination with the Matlab Simulink communications toolbox [9] 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] S. Zander, S. J. Murdoch, "An Improved Clock-skew Measurement Technique for Revealing Hidden Services," in *Proceedings of Usenix Security*, July/August 2008.

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

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

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

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

[11] 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.

[12] 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

	Intermed. host	**Statistic (W)**	**p-value**
Day (100%)	1	0.973	$\ll 1\%$
Day (96%)	1	0.995	2.6%
Night (100%)	1	0.987	$\ll 1\%$
Night (96%)	1	0.996	1.5%
Day (100%)	2	0.992	0.05%
Day (96%)	2	0.982	$\ll 1\%$
Night (100%)	2	0.981	$\ll 1\%$
Night (96%)	2	0.991	$\ll 1\%$