Data Exfiltration from Network-Isolated Virtual Computing Systems with Infrastructure Mediated Power Analysis

Zechariah D. J. Wolf, Eric C. Larson, and Mitchell A. Thornton(✉)

Darwin Deason Institute for Cybersecurity at Southern Methodist University, Dallas, TX 75205, USA
{zechw,mitch,eclarson}@smu.edu

Abstract. Side channels and covert channels have proven to be dangerous avenues for the leakage of sensitive information from computing systems. In this work, we propose and perform an experiment to investigate side and covert channel possibilities in virtual, enterprise environments. Using a power analysis approach, traces of the power signal from an enterprise-level server rack are examined in the frequency domain, revealing that electrical power line fluctuations can be correlated with patterns of system load on a virtual machine (VM) hosted by a server on the rack. We show the potential for a malicious insider to use this approach to establish a covert channel for the exfiltration of sensitive data from a VM. By encoding binary information into the signature of the spectral content of the power signal while the system experiences varying degrees of CPU load, it is shown that communication using binary symbols is feasible. The noise resilience of the channel is evaluated, and it is shown that communication at a rate of at least 1 bit per second is possible using this method.

Keywords: Covert channel · Power analysis · Network isolation

1 Introduction

A common term in the field of security is network isolation. Network isolation is the practice of disconnecting a computer from any external, insecure networks. Sometimes referred to as "air-gapped," a network isolated machine is generally understood to be more secure than a machine connected to a wider, possibly insecure network. Computing systems are air-gapped in order to enhance security for a device which may be housing highly sensitive data, by reducing the potential attack surface, i.e. cutting off the possibility for infiltration from external attackers. While this is certainly an advantage of practicing network isolation, security conscious individuals or companies should avoid understanding network isolation as a holistic security solution. As is demonstrated in this paper, even in highly secure, "air-gapped" and virtualized computing environments, the potential exists for data exfiltration by malicious insiders.

G. Lenzini et al. (Eds.): ICISSP 2023/ICISSP 2024, CCIS 2459, pp. 161–181, 2026.
https://doi.org/10.1007/978-3-031-89518-0_8

1.1 Side and Covert Channels

Side channels and covert channels are important concepts to understand in terms of data leakage from a computing system. Generally speaking, a side channel is a medium by which information about the state of a computing system is unintentionally present. If an attacker has access to a side channel, they may be able to gain unauthorized access to sensitive data through analysis of the side channel, unbeknownst to the victim. Well-known examples of side channels include information streams such as power consumption measurements and electromagnetic emanations from a computer during operation. In a side channel attack, a malicious party leverages side channel data to capture sensitive information from a computing system [19].

Related to the concept of a side channel is a covert channel. While a side channel can be used by an attacker to gather information from a system unbeknownst to the victim, a covert channel can be established when a source of information leakage is used deliberately to communicate information by encoding it over a channel that was not meant for communication [5]. Covert channels are an important area of research because they provide a means of communication outside the jurisdiction of organizational security policies. Thus, they can be used to transmit sensitive information to unauthorized parties. An effective covert channel, by making use of unconventional means of communication, can be used to leak information while evading detection.

1.2 Covert Channels in Network Isolated Systems

The focus of this work is to investigate, from a defensive perspective, what avenues of data leakage would still be available to malicious parties even in an air-gapped and virtual environment. Using an experimental apparatus consisting of an enterprise level server rack, it is demonstrated that it is possible to establish a covert channel via power analysis. Our power analysis approach involves capturing traces of the server's electrical power signal, while a VM generates patterns of system load on one of the servers in the rack. We show that these patterns of system load have a distinct signature in the frequency domain of the rack's power signal, which would enable a malicious insider to use a binary communication scheme to exfiltrate sensitive data from the VM.

This investigation represents the continuation and extension of research which we presented in a paper at the 9^{th} International Conference on Information Systems Security and Privacy [22]. This original paper documented the results of our early research, identifying the side channel of the server's power signal and its potential for use as a covert channel. As an extension of this work, in this paper, we perform extensive additional experimentation to evaluate the noise resilience of binary communication over this power-analysis-based covert channel. In particular, we investigate the use of classification and the hyper-parameters of the classification thresholding that yield accurate results across a range of background noise.

2 Power Analysis Background

Power analysis research took off with Paul Kocher's experiments in 1999, which popularized the concept of using device power information as a side channel to extract

sensitive information from a computing process. Kocher et al. [11] were focused on the context of encryption algorithms, showing that the hardware on which the encrpytion is performed, leaks information about the data being processed via its power consumption. By recording the power consumption of the computing circuitry during encryption, their research showed that it is possible to extract Data Encryption Standard (DES) keys. Kocher et al. identified that these kinds of power analysis side channel attacks are made possible by the nature of the transistors in the computer's logical circuitry, which consume power when their binary state changes. As such, these kinds of attacks can be difficult to protect against. More recent examples of power analysis attacks on encryption algorithms, such as [12,15], show that power analysis remains a relevant area of research in the field of security.

This type of power analysis research has broad applications in the field of side channels. In this work, we apply the concept of power analysis to investigate the potential for extracting information from the power consumption information of a server rack under operation. In particular, we look at the electrical power line fluctuations (EPLF) which is generated by a server computer using a switched mode power supply (SMPS). This EPLF has been used by a number of researchers for identifying electrical devices and appliances [7,16] for identifying changes in electrical device signature that can be repurposed for interaction techniques [2,6], and has been shown to leak information regarding television programming [4]. This type of power sensing is typically described as "infrastructure mediated" because it relies on a single sensing point in a building's power infrastructure. Thus, the EPLF sensed from the server is observable along numerous outlets on the power line. We show how this EPLF might be leveraged by a malicious insider, to exfiltrate information from a VM on the server rack.

3 Server Rack Setup

In order to test the viability of an EPLF-based covert channel, we make use of a custom-built, enterprise level server rack. This rack houses several servers, running a bare metal hypervisor operating system. The power supplies for each server, as well as the network switches and other rack components, use the rack's power distribution unit (PDU) that is connected to a 2U 120 VAC power supply that services the entire rack. This server rack is, by design, network isolated, i.e. its management network is self-contained, and not connected to any wider insecure networks. The rack is depicted in Fig. 1.

3.1 EPLF Collection Equipment

In order to record measurements of the power consumption, the rack is plugged into a surge protector, and our customized power line interface (PLI) module is plugged into a nearby power outlet on the same circuit breaker. This PLI module is a circuit which acts a high pass filter, and was constructed based on a design by Gupta et al. [7]. The filtration effect of this module allows us to collect the high frequency voltage noise which we expect to be generated by the SMPS of the various servers on the rack. The PLI module is then connected to an oscilloscope (Picoscope 4000a series), which is used to record traces of the rack's electrical power signal via a USB connection to a laptop. The full EPLF collection equipment is summarized in Figs. 2 and 3.

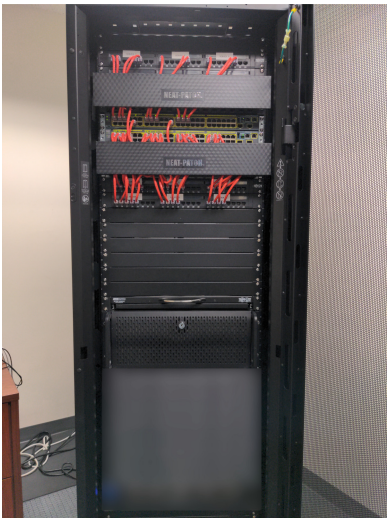

Fig. 1. A picture of the server rack used for power analysis experiments.

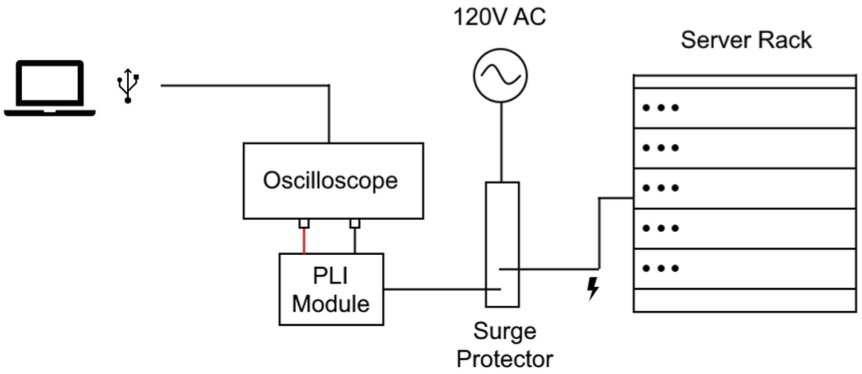

Fig. 2. Experimental apparatus for monitoring electrical power signal.

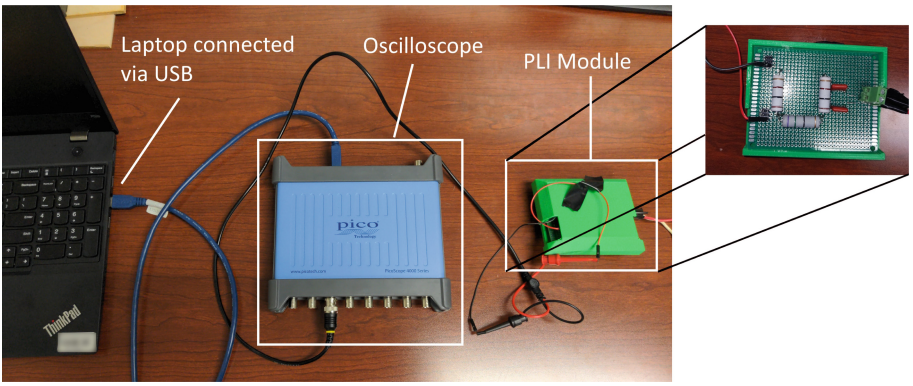

Fig. 3. A picture of the experimental apparatus, featuring the PLI module, oscilloscope, and laptop.

4 Covert Channel Simulation Procedure

The purpose of this experimental apparatus is to collect traces of the rack's electrical power signal, with the goal of analyzing any correlations between the characteristics of the signal, and system activity on a server. For this purpose, we spin up a Linux VM on one of the servers in the rack. As a data exfiltration scenario, we envision a malicious insider with access to sensitive information on the VM. We hypothesize that the signature of the rack's power signal should be characteristically different when the VM is under high load, versus when it is at idle. If this is the case, it would facilitate a binary communication scheme, in which binary ones and zeros are encoded into the binary (high load or low load) state of the VM, which can be extracted from the recorded power signal.

4.1 Generating System Load Patterns

In order to achieve these states of high and low load on the VM, we use an open source Linux program called `stress-ng` [3]. `stress-ng` is a command line program which can be used to generate varying levels of stress on a system. In this case, we use it to generate system load on the VM with a target CPU utilization percentage. The goal is to generate a load pattern that, in turn, causes a corresponding variation in power consumption that is recognizable and distinguishable, even in the presence of background noise. To accomplish CPU load variations, a shell script is used to initiate a `stress-ng` load for a certain time interval, and then to sleep for a subsequent interval. The shell script thus serves to modulate the power consumption profile that varies in proportion to the CPU load changes. Therefore, the shell script modulator creates states of high and low power consumption of varying respective duration that are observable as a power analysis side channel. A particular binary string corresponds to the different duration times of the high and low load states within the script that can be observed by monitoring the 120 VAC power line serving as a side channel for exfiltrating the binary stream of information. For the CPU utilization amount, we use 100% load as the target, in an attempt to create the starkest difference between the idle and load states of the VM. The result of using the shell script modulator is that the VM experiences a sort of CPU utilization square wave, where the utilization (and transitively, the power consumption) sharply transitions between near 0% and near 100% at a regular period.

4.2 Initial Testing

The first step in our experimentation was to observe the power signal in the frequency domain, to see if there was any characteristic change in the power signal as the VM modulated between states of high and low load. This is achieved using a spectrum analyzer (included with the oscilloscope unit) to monitor the frequency spectrum of the signal in real-time. Figures 4 and 5 show the spectrum analyzer view as captured in real-time during power consumption modulation. The frequency spectra shown in the figures range from 0–100 kHz.

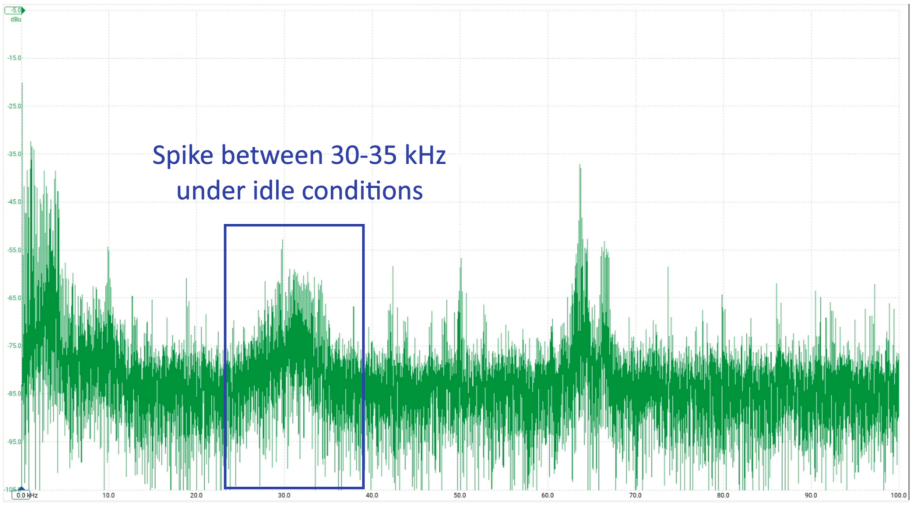

Fig. 4. Real-time frequency spectrum during idle conditions showing voltage amplitude (dBu) versus frequency (kHz) [22].

Figure 4 shows the spectrum analyzer view when the VM is idling. With no high load tasks running, the frequency spectrum shows a noticeable spike in the range of 30–35 kHz. Importantly, this spike is not present when the VM initiates a `stress-ng` load at 100% CPU utilization, as depicted in Fig. 5. This change was very easily noticeable by human observation of the spectrum analyzer, as the spike rose and fell, in time with the high load and idle states of the VM. As an initial result, this was strong evidence that this EPLF method might have potential for use as a covert channel, in which binary information could be represented by the load level of the VM.

4.3 Data Collection and Processing

To further assess the possibility for covert communication using this method, we proceeded to record traces of the power signal from the oscilloscope under varying system conditions. Because the spike is noticeable in the 30–35 kHz range, a sampling rate of at least 70 kHz is necessary to exceed the Nyquist rate for this phenomenon. Thus, we chose to sample the power signal at a rate of 100 kHz. For the system conditions, we collected several traces while the shell script modulator was running, to capture simulated covert communication at varying data bit rates. We also captured traces of the system in a static low or high load state, in order to better characterize the frequency spectrum of the signal in either state. A Python script, based on a script provided by the maintainers of the Picoscope SDK package [13], is used to collect the traces on a USB-connected laptop.

The traces are captured in the time domain, but our analysis is focused on the frequency domain characteristics of the signals. Because these characteristics are changing over time as the load levels on the VM change, we use a short-time Fourier transform

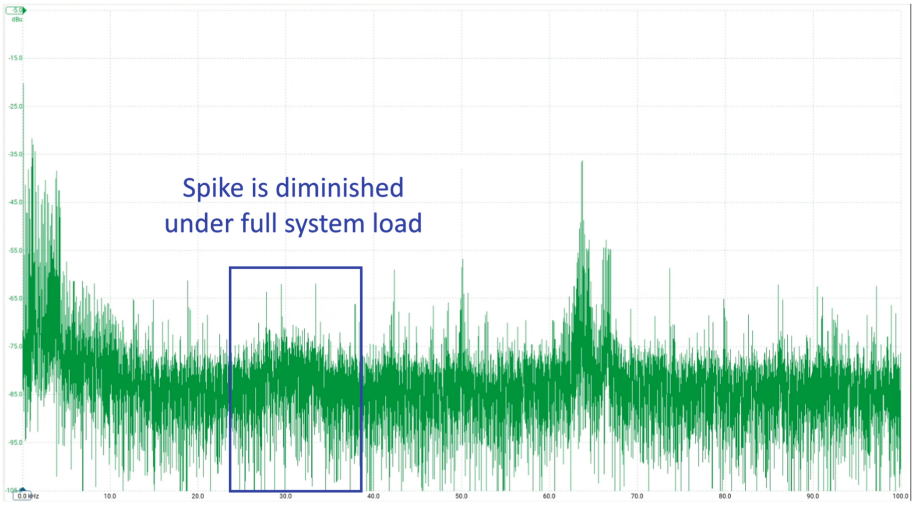

Fig. 5. Real-time frequency spectrum during load conditions showing voltage amplitude (dBu) versus frequency (kHz) [22].

(STFT) to analyze the spectral content of the signal as it shifts over time. The STFT works by breaking the time domain signal down into shorter, overlapping segments, and taking the Fourier transform of each segment. The result is a complex-valued matrix where each column represents the Fourier transform of the respective segments. A magnitude spectrogram is used to visualize the magnitudes of each value in the matrix. A magnitude spectrogram is a visualization showing three pieces of information: frequency (vertical axis), time (horizontal axis), and magnitude (color intensity). If covert communication is feasible using this method, we should expect to see high and low magnitude bands in the 30–35 kHz region, as the system state alternates between high and low load. The signal processing procedure is performed using open-source packages in Python. For signal processing algorithms and spectrogram visualizations we use Librosa [14] and SciPy [20]. The visualizations are supported by Matplotlib [9] and Seaborn [21]. Finally, we make use of Numpy for representing the arrays of data during processing [8].

4.4 Simualated Covert Channel Spectrograms

The first trace of interest captured the power signal while the shell script modulator operated with a period of one second per system state (high load or idle). This would correspond to the encoding of binary information at a rate of 1 bit per second. To process the signal, we used an STFT with a 2048 point Hamming window (about 20 ms), 75% overlap. Figure 6 shows the resulting spectrogram. Notice that the magnitude is reported in decibel-Watts (dBW), in order to reduce the dynamic range of the values.

The spectrogram in Fig. 6 shows very clearly that it is possible to encode binary information into the power signal using the high and low load states on the VM. More-

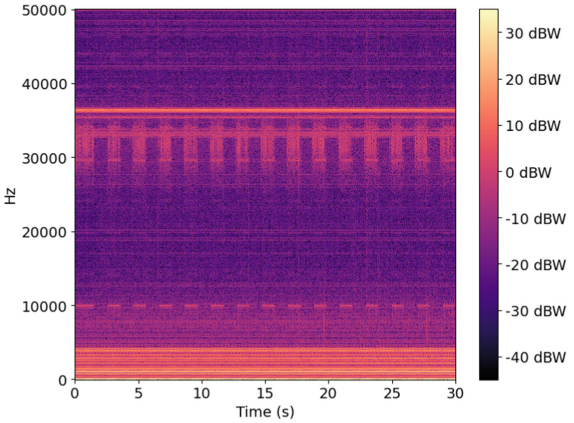

Fig. 6. Magnitude spectrogram for covert channel sending symbols at a rate of 1 bit per second [22].

over, it is possible to read that information simply by human observation of the frequency spectrum of the power signal, as visualized in a spectrogram. The high load and idle states are easily identified by the regions of low and high magnitude respectively, in the 30–35 kHz region in the spectrogram. Interestingly, there is also a noticeable spike in magnitude at 10 kHz, although the frequency range of this phenomenon is not as wide. Both of these regions of interest in the frequency spectrum might be useful for quantitatively differentiating between the two system states. As an additional experiment, we also recorded a trace of the power signal with a shell script modulation period of five seconds. This would correspond to a data bit rate of 0.2 bits per second. The spectrogram for this trace is shown in Fig. 7.

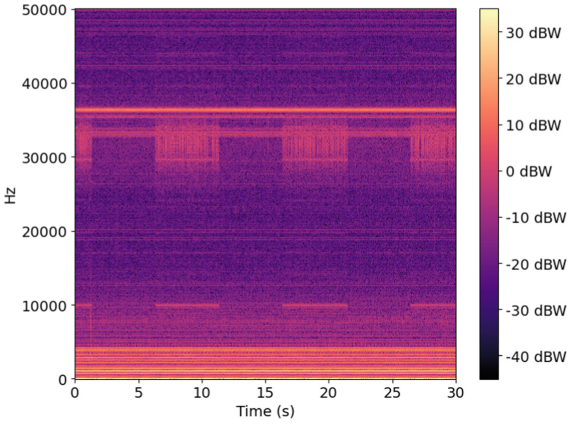

Fig. 7. Magnitude spectrogram for covert channel sending symbols at a rate of 0.2 bits per second [22].

In Fig. 7, the clear regions of high and low magnitude, corresponding with the VM's CPU utilization changes, once again show that binary covert communication using this method is viable.

4.5 Ternary Data Symbols

Aside from changing the length of time per system state, another option to increase the rate of information transfer over this channel is to use higher-radix data symbols. In our experimental setup, we test a ternary data symbol communication scheme, in which three levels of CPU utilization are used to represent three distinct data symbols. This is accomplished by modifying the shell script to cycle the CPU utilization between idle, 50%, and 100%. Ideally, this should create three discernible states recognizable as data symbols from observation of the power signal's spectral content. To test this, a 30 s trace of the signal was recorded while the shell script modulator alternated between the three states. Figure 8 shows the resulting spectrogram.

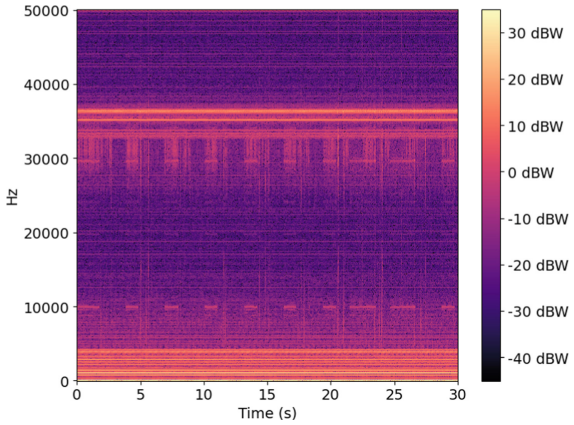

Fig. 8. Magnitude spectrogram for covert channel sending ternary data symbols [22].

The spectrogram for the ternary case is not as easily interpretable as the binary case by simple human observation. The magnitude values in the frequency ranges of interest are not clearly discernible as low, medium, and high states. Even so, it may be the case that a computational approach for state recognition—a machine learning model for instance—may be able to classify these three system states. Further experimentation would be required to assess the feasibility of this ternary communication scheme.

5 Binary Information Exfiltration

The magnitude spectrograms show that it is possible to encode information from the VM by modulating the CPU utilization and observing the system's EPLF, to extract

binary data symbols. If a malicious insider were to implement this covert channel for data exfiltration, they would need a method to decode the binary information from traces of the power signal. This could be done by classifying the observed EPLF of the system at a given time as either a 0 or 1. Thus, the next step in our analysis is to devise such a classification model, and evaluate its performance under varying conditions. For this purpose, we use an ROC curve to determine classification thresholds, and measure the area under the ROC curve (AUC) and classification accuracy as performance metrics. We also evaluate the resilience to noise by simulating noisy conditions using additive white Gaussian noise (AWGN). Finally, we perform a hyperparameter search to optimize the classifier to operate under the most noise resilient classification framework.

5.1 Classification Task Formulation

In the practical application of a covert channel, a classification model would need to classify segments of a captured signal as either a low power consumption state, or a high power consumption state as affected by the CPU utilization of the VM. These high and low states will be represented with labels of 0 and 1, representing the data bits present within the channel.

The length of the signal segment per data symbol could be variable depending on the application, and levels of background noise. For generalization purposes, we define a sample as one column of the STFT matrix. Each column of the STFT is representative of the spectral content of the signal over a small interval of time. The length of time represented by each column is dependent on the time resolution of the STFT, determined by the window length and percent overlap used in the STFT. Under this framework, the high and low states of the system would be recognized by the signature of their spectral content.

5.2 Samples and Scoring

First, the raw signals in the time domain are used to create an STFT matrix. For initial testing, the free parameters of the STFT are kept the same as described previously for the creation of spectrograms. Given two STFT matrices, one for the idle signal and one for the load signal, we can extract samples of either class, and use an ROC curve to evaluate the classification performance at varying classification thresholds.

Each column of the STFT is labeled as either 0 or 1, based on the signal from which the STFT is computed. The second piece of information needed for an ROC curve is a score for each sample, which determines the predicted class based on a given scoring threshold for dividing the classes. For this purpose, we develop a methodology for scoring samples based on their spectral content.

5.3 Computing a Spectral Content Signature

In order to achieve this objective, we compare the spectra of the idle and load signals and determine the frequency ranges of interest where the signals are most distinct in the

frequency domain. This process is essentially a computational approach for what was observed in Figs. 4 and 5, where it is possible to visually distinguish the system state by human observation of the spectrum analyzer display.

This can be achieved computationally by taking a small sample (5%) of the traces that capture the system in a static state (idle or load). From each of the samples of these traces, we compute the STFT, logarithmically scale the result, and average the magnitudes across the time frames of the STFT. The result is a single spectrum, representing the average logarithmically scaled magnitudes of the Fourier coefficients from each time frame (column) of the STFT. When this process is accomplished for the signals for each system state, the results can be subtracted to produce an array of the difference (measured in dB units) between the two spectra. From this array of differences, we select the frequency bins to use for classification if the difference surpasses a given difference threshold specified with a dB unit that serves as a hyperparameter of the classification process. This process is visualized in Fig. 9. Once the frequency bins have been selected, the remaining 95% of the time domain signals are used for assessment of classification performance.

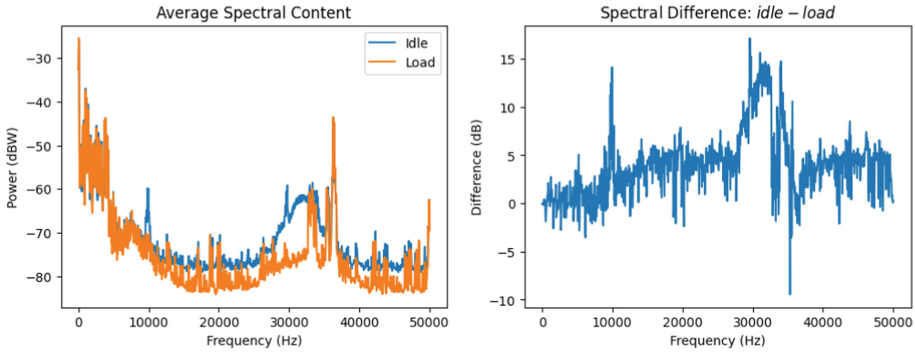

Fig. 9. Comparison and subtraction of mean spectral content from the STFT.

Figure 9 shows, on the left, the comparison of spectral content between the idle and load signals by averaging across the time frames of the STFT. This corresponds roughly with the real-time spectrum analysis view. On the right is displayed the subtraction of the averaged spectra. This difference plot visualizes the regions of the signal in the frequency domain where the spectral content is most distinct. As could have been expected from the spectrum analyzer snapshots in Figs. 4 and 5, the difference is, once again, most pronounced at 10 kHz and 30–35 kHz. Given the difference between the two signals at a set of frequency bins where the difference exceeds a chosen threshold value in dB, the mean value of the frequency bins should summarize the distinction in a singular value. Thus, the score is defined as the mean magnitude in dBW of the selected frequency bins. For assessing classification performance in this task formulation, we use an ROC curve to test different thresholds in the dynamic range of the score as classification boundaries, to get an understanding of the generalized performance of a classification model on this data.

6 Evaluating Noise Resilience

Since the signals are quite distinct in the frequency domain at certain ranges, it would not be unexpected for the ROC curve to show near-perfect performance under the relatively noise-free testing environment. The data for this experiment is collected under ideal conditions with minimal background noise obfuscating the signal. In a practical application of a covert channel however, there would likely be significant background noise from other VMs running on the same host machine, or different host machines in the server rack that share the same power source. Accordingly, this investigation would be incomplete without attempting to quantify the classification process' resilience to noise.

For this purpose, we use AWGN to simulate the addition of noise to the signal. By adding a random noise vector to the signal in the time domain that conforms to a Gaussian distribution to simulate white noise, we can observe the decline in classification performance as the SNR decreases. This observation will provide an idea of how effective the covert channel would be under more realistic, noisier conditions. It also provides a point of comparison between the classification methodologies.

Quantitatively, there are two metrics that we use to measure the performance under noise conditions: area under the curve (AUC) of the ROC curve, and accuracy. Accuracy is calculated by using the classification threshold of the equal error rate to predict a 0 or 1 for each sample, and computing the percentage of correct predictions. For ROC analysis and metric calculations, we use an open source machine learning package in Python, `scikit-learn` [17].

6.1 Noise Results with Static STFT Parameters

As outlined above, the classification procedure uses the mean of the magnitude in dBW of the Fourier coefficients at the frequency bins of interest as a sample. We test a range of SNR values from 20 dB down to –5 dB.

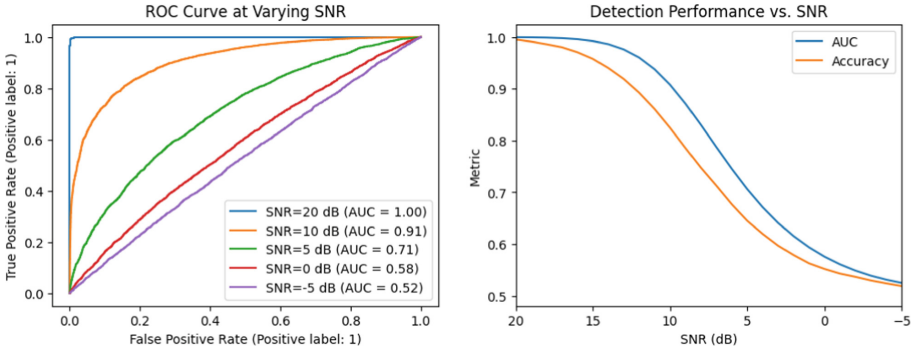

Fig. 10. Classification metrics for initial STFT parameter selection with AWGN.

Figure 10 shows the ROC curves for increasing noise levels on the left. On the right, the AUC and accuracy (vertical axis) are plotted against decreasing SNR on the

horizontal axis. These plots visualize the decaying classification performance as noise increases. From a visual point of view, a more noise resilient classification method should show a higher-valued knee for the ROC curves and a more gradual downward slope for AUC and accuracy against decreasing SNR. The ROC curves show that the classification is near-perfect until the SNR reaches about 20 dB. As the SNR decreases past 20 dB, the performance deteriorates, and the AUC score reaches random chance levels (0.52) by –5 dB. Given this promising initial result, we investigate methods for optimizing the classification performance under noisy conditions.

6.2 Noise Results Using DPSS Windowing

One method for improving performance could be to apply windowing in the frequency domain in an attempt to reduce the effect of noise. For this purpose, we use a multi-tapering approach with the discrete prolate spheroidal sequences (DPSS) [18]; i.e., a specific sequence of window functions designed to reduce STFT noise. The classification procedure is similar using DPSS, except the STFT is transformed by multiplying each complex-valued column by several DPSS tapers, and taking the mean of the products. The DPSS tapers are computed according to the following free parameters in Table 1:

Table 1. Parameter selection for DPSS windowing.

Parameter	Value
Window Length	2048
Standardized Half Bandwidth (NW)	500.0
Number of Windows (K_{max})	5

Each column in the STFT is multiplied element-wise by the first K_{max} tapers in the DPSS sequence, to produce K_{max} products. The mean of the products across each frequency bin is taken to produce a single complex-valued vector of Fourier coefficients. The mean of the logarithmically-scaled magnitudes is understood as a sample in this method. After the DPSS pre-processing is complete, the same noise resilience testing procedure is applied. Figure 11 shows the classification performance with DPSS windowing in the frequency domain.

Initially, these results seem to indicate that using DPSS does not seem to have a significant impact on the classification performance. To confirm this indication, we observe the weighted averages of the AUC scores, using weights that increase linearly with SNR, as a measure of noise resilience. After performing classification with and without multi-tapering, the weighted AUC score is calculated to be equal for each method, with a value of 0.63 for a SNR range of 20 dB down to –5 dB. Thus, it can be concluded quantitatively in this scenario that multi-tapering with DPSS does not offer any significant improvement in terms of noise resilience.

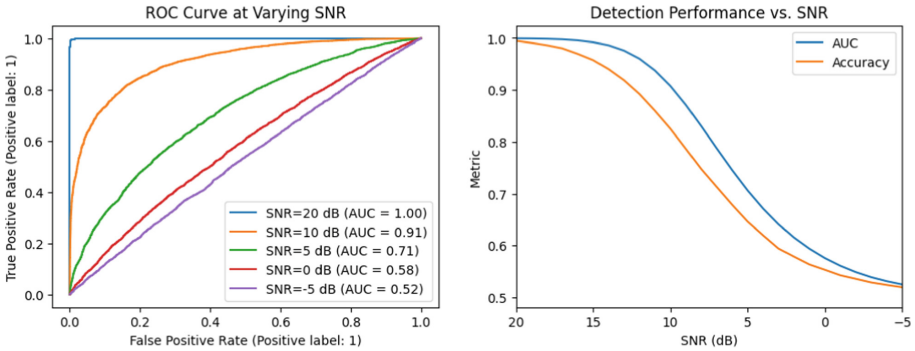

Fig. 11. Classification metrics for initial STFT parameter selection with DPSS windowing applied.

7 STFT Parameter Optimization

In addition to applying multi-tapering in the frequency domain with DPSS, another approach for potentially improving classification performance is adjusting the parameters of the STFT. The methodology so far has assumed a static free parameter selection for the STFT. However, there are different approaches in terms of window length, window type, and overlap percentage that are worth investigating as well to see if there is an effect on classification performance.

Moreover, these parameters have a direct effect on the frequency resolution, and time resolution of the STFT, which in turn affects the theoretical data bit rate of the channel. For these experiments, we match the window length to the FFT (fast Fourier transform) length, i.e. the number of Fourier coefficients which are computed from segments of the time domain signal. For this reason, a larger window length equates to a more precise frequency resolution, since each frequency bin represents a smaller portion of the frequency range of the signal. However, this also means that the time resolution and the frequency resolution of the STFT are inversely correlated; as the frequency resolution becomes more precise, the time resolution becomes less precise, and vice versa. This occurs since a larger window length means that the signal is broken up into larger segments in the time domain, so the resulting time frames (columns) of the STFT represent larger intervals of time.

Because of this phenomenon, we can expect a trade-off between noise resilience, and the theoretical data bandwidth of the covert channel as consistent with Shannon's SNR versus data rate theorem. That is to say, a better frequency resolution should result in superior classification performance at lower SNR, since there are more frequency bins to use in differentiation of the system states. On the other hand, a better time resolution results in a better theoretical data bandwidth. For this experiment, we define the theoretical data bandwidth as the maximum bit rate, measured in bits per second, that could be possible within the covert channel, given the time resolution of the STFT. This is calculated based on the window length, and overlap percentage. Given a window length N, a percent overlap p, and a sampling frequency f_s (the sampling frequency is

100 kHz throughout all trials), the theoretical data bandwidth of the channel, in bits per second, is calculated as shown in Eq. 1.

$$Bandwidth = \left[\frac{N}{f_s}(1-p)\right]^{-1} \tag{1}$$

Thus, when optimizing the free parameters, we have two inversely proportional objective functions to consider: data bandwidth, and classification performance (AUC). In the following section, we provide details of our procedure for testing hyperparameter selections in order to observe the trade-off between data bandwidth and classification performance. After optimization, we use the optimal parameters for classification performance to assess the overall noise resilience of this channel.

7.1 Procedure

In Python, we make use of a package called Optuna [1] for the purpose of parameter optimization. Similar to the previous tests using AWGN, we assess the classification performance as noise increases (SNR decreases). The most noise resilient model is the one whose performance deteriorates the least as the SNR decreases. We quantify this by classifying the system state at different noise levels for each set of parameters. Then the classification performance is summarized by taking the weighted average of the AUC scores over the SNR range. The weights increase linearly as the SNR decreases, so that the AUC for lower SNR values factors more heavily into the overall score.

While the classification performance is determined by experimentation with simulated noise, the theoretical data bandwidth is deterministically calculated using a given set of STFT hyperparameters, as shown in Eq. 1.

When optimizing a set of hyperparameters, Optuna performs an automated analysis, in which some number of trials is performed, each trial representing a selection of values for the hyperparameters. When finished, the value of the objective function is reported for each trial. When all trials have completed, the best hyperparameter values can be determined by examining the trials with the highest objective function scores. In our case, a trial consists of the following steps:

1. Select test values for the hyperparameters
2. Determine the frequency bins that will be used for classification
3. For each SNR in the range of 10 to –20 dB (increments of 1 dB):
   (a) Add noise to the time domain signals to achieve the given SNR
   (b) Take the STFT of each signal and score each time frame by the mean of the magnitudes in dBW at the selected frequency bins
   (c) Report the AUC for this SNR value
4. Report the weighted average of the AUC scores for all SNR, and the theoretical data bandwidth for this selection of hyperparameter values

7.2 Hyperparameters

Aside from the parameters of the STFT, another hyperparameter that may affect classification performance is the difference threshold at which frequency bins are selected for

inclusion in the calculation of the mean spectral content. As was portrayed in Fig. 9, the selected difference threshold value is determined by averaging across the time frames of the STFT for a sample of the idle and load signals, and subtracting the two spectra, which produces an array of the differences between the two signals in dB. If the difference in dB is greater than or equal to the difference threshold, than the frequency bin represented at that index is considered in the computation of the classification score. Thus, the difference threshold may be an important hyperparameter to consider for optimization.

In summary, the four hyperparameters that we optimize over are: window length, window function type, overlap percentage, and difference threshold. Each hyperparameter is assessed with a different range of possible values in order to find the most optimal combination.

Table 2. Summary of hyperparameters and the range of values used for testing.

Hyperparameter	Possible Values
Window Length	$[2048, 131072]$
Window Type	$\{Hann, Hamming, Blackman\}$
Overlap Percentage	$\{0\%, 25\%, 50\%, 75\%, 90\%\}$
Difference Threshold	$\geq$1 dB

7.3 Optimization Results

After performing a study consisting of 200 different selections of hyperparameter values, we found that the most important factor is the window length. In Optuna, the parameter importances can be calculated using fANOVA [10], a method for evaluating the importance of a model's hyperparameters. Figure 12 shows the importance of the parameters relative to the weighted average of the AUC scores.

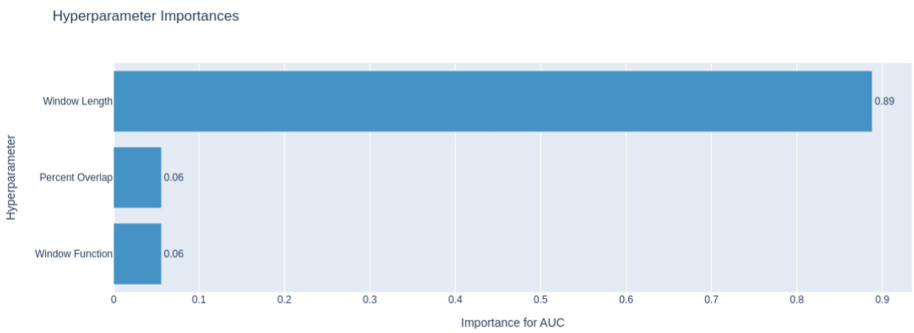

Fig. 12. Hyperparameter importance scores according to fANOVA.

This result confirms that window length is by far the most important factor. Window function and overlap percent are not completely unimportant, but their impact on classification performance is not as significant as the window length. This is further confirmed by examining the Pareto frontier plot for the study. The Pareto frontier shows the result of each trial in terms of its theoretical data bandwidth, and classification performance.

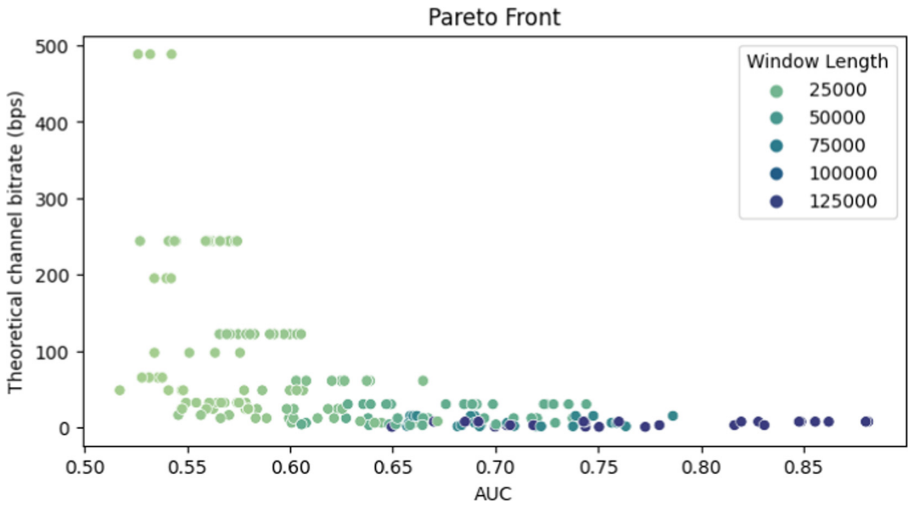

Fig. 13. Pareto frontier plot showing theoretical bit rate vs. weighted AUC score. Each trial is colored by its window length.

Figure 13 shows the Pareto frontier plot, with the trials colored by their window size. The trend is clear: a larger window size results in better classification performance. Trivially, a smaller window size results in a better data bandwidth (this was computed deterministically), which comes at the cost of classification performance. The best hyperparameter values for maximizing the objective functions (weighted AUC and theoretical bit rate), are the points which reside on the Pareto frontier, i.e. the points on the curve which are not dominated by any others [1]. The hyperparameter values on the Pareto frontier are shown in Table 3.

Judging by classification performance, the best set of parameters found in the study are summarized in Table 4.

Using this parameter selection, we can visualize the classification process relative to noise levels, to get a better understanding of the best possible noise resilience. Figure 14 shows the ROC curves and the AUC/accuracy plotted against decreasing SNR.

The AUC and accuracy are initially near perfect, gradually decreasing until the SNR reaches about –7 dB. At this point, the accuracy score starts decreasing sharply. The AUC starts trending downwards as well around –10 dB, and both metrics reach random chance levels by –30 dB. This means that with these parameters, a malicious insider

Table 3. Hyperparameter values for the trial runs on the Pareto frontier.

Window Length	Window Type	Overlap Percentage	Difference Threshold (dB)	Weighted AUC	Theoretical bit rate (b/s)
2048	Blackman	90%	8.5	0.542	488
4096	Hamming	90%	9.5	0.574	244
8192	Hann	90%	16.0	0.605	122
16384	Hamming	90%	17.5	0.665	61.0
32768	Hann	90%	22.5	0.744	30.5
65536	Hann	90%	20.5	0.786	15.3
131072	Hamming	90%	17.5	0.881	7.63

Table 4. Optimal hyperparameter values.

Hyperparameter	Best Value
Window Length	131072 (2^{17})
Window Type	Hamming
Overlap Percentage	90%
Difference Threshold	17.5 dB

that is exploiting a covert channel for unauthorized data exfiltration could expect near-perfect symbol detection accuracy up to an SNR of about –8 dB. After this point, accuracy would decrease sharply, but it would still exceed random chance classification until around –30 dB SNR. It may be possible to achieve even better noise resilience using larger window sizes, however, because we only tested a 1 s period per system state in our experiments, at this time it makes sense to evaluate the noise resilience of the covert channel when the window size is close to this one second period (a window size of 131072 is about equal to 1.3 s at our sample rate of 100 kHz).

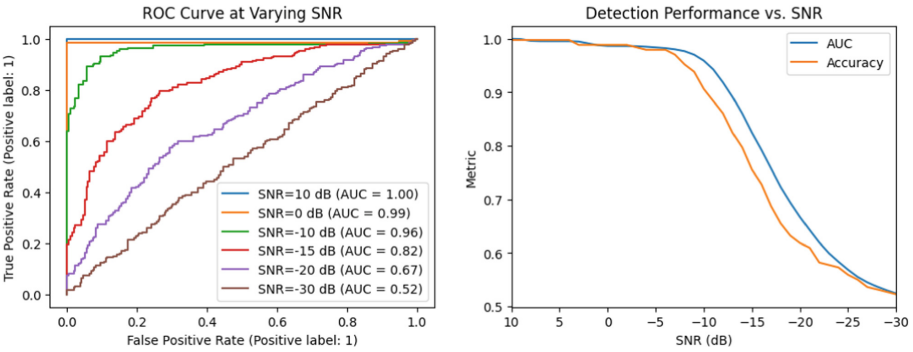

Fig. 14. ROC curves and classification metrics vs. SNR for optimal hyperparameters.

8 Conclusion

With the parameters optimized for noise resilience, we found that the maximum theoretical bit rate for communication would be about 7.63 bits per second, as determined by the time resolution of the STFT. This number represents the maximum rate of noiseless communication a malicious insider would be capable of, assuming it was possible to alternate between systems states over seven times per second. Future experimentation would be required, however, to analyze just how quickly the system could alternate between recognizable states of high and low power consumption in a way that is interpretable by analysis in the frequency domain as we have shown thus far. This would provide a better estimate of the maximum rate of covert communication using this channel. At this time, however, we can conclude that it is possible to exfiltrate binary information using this method, at a rate of at least 1 bit per second, with a high degree of resilience to background noise, such that greater than 95% accuracy when recognizing data symbols would be possible even at an SNR of –8 dB.

8.1 Data Capture Constraints

Side Channel Mechanism. One topic that warrants further investigation is the mechanism that causes this side channel. It is our hypothesis that the explanation for why this covert channel security exploit is viable lies in the response of the SMPS to increased power consumption on the server. The power supplies for the servers in the rack are equipped with cooling fans, which are designed to dissipate heat from the components in the power supply unit. When the VM increases its CPU utilization, it should cause the power supply's fan array to receive increased power, in order to dissipate the excess heat. We hypothesize that what is enabling this exploit is a rapid change in the fan's speed in response to the system load on the VM. The spike observed in Fig. 4 is likely the fan's signature in the frequency domain. When the spike "dissipates," we speculate that it is actually moving to a higher frequency (beyond the Nyquist frequency) due to the increased power draw as the fan array receives more power to increase its speed. More specifically, because a sampling rate of 100 kHz was used to capture all of the traces in these experiments, we cannot detect any phenomena in the frequency domain whose frequency is higher than 50 kHz—the Nyquist frequency for our sampling rate. If this hypothesis is correct, the sampling rate used in our experiments was not sufficient to capture the spike moving to a higher frequency location in the spectrum. A future experiment could test this hypothesis by sampling the power signal at a higher sampling rate to see if the spike from higher power consumption becomes observable.

Experimental Background Noise Conditions. While our experiments show promise as a proof of concept, our test cases represent somewhat ideal circumstances that may not be typical of an enterprise computing environment. We performed an initial investigation of this limitation by simulating noise conditions with additive white Gaussian noise. While these simulations produced interesting and promising results, they only offer a prediction of how the covert channel could perform under more realistic circumstances. Future research could take the next logical step and evaluate the feasibility

of the channel under practically implemented background noise conditions, by powering up and stressing other virtual machines within the rack during a communication session using the covert channel to simulate normal conditions. These results could be compared to our predictions based on noise simulation.

We have performed initial testing with some background noise conditions, but not enough data was collected for a full evaluation. Thus, a more extensive exploration should be done before drawing conclusions about the effectiveness of using this side channel for data exfiltration in a typical enterprise computing environment with multiple servers running. However, at the present time, our experiments and simulations indicate that this method poses a potential security risk for network-isolated, enterprise computing systems, and warrants further investigation.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
2. Chen, K.Y., Cohn, G.A., Gupta, S., Patel, S.N.: uTouch: sensing touch gestures on unmodified lcds. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2581–2584 (2013)
3. ColinIanKing: stress-ng (2023). https://github.com/ColinIanKing/stress-ng
4. Enev, M., Gupta, S., Kohno, T., Patel, S.N.: Televisions, video privacy, and powerline electromagnetic interference. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 537–550 (2011)
5. Freiling, F.C., Schinzel, S.: Detecting hidden storage side channel vulnerabilities in networked applications. In: Camenisch, J., Fischer-Hübner, S., Murayama, Y., Portmann, A., Rieder, C. (eds.) SEC 2011. IAICT, vol. 354, pp. 41–55. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21424-0_4
6. Gupta, S., Chen, K.Y., Reynolds, M.S., Patel, S.N.: Lightwave: using compact fluorescent lights as sensors. In: Proceedings of the 13th International Conference on Ubiquitous Computing, pp. 65–74 (2011)
7. Gupta, S., Reynolds, M.S., Patel, S.N.: Electrisense: single-point sensing using emi for electrical event detection and classification in the home. In: Proceedings of the 12th ACM International Conference on Ubiquitous Computing, UbiComp '10, pp. 139-148. Association for Computing Machinery, New York (2010). https://doi.org/10.1145/1864349.1864375
8. Harris, C.R., et al.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). https://doi.org/10.1038/s41586-020-2649-2
9. Hunter, J.D.: Matplotlib: a 2D graphics environment. Comput. Sci. Eng. **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
10. Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient approach for assessing hyperparameter importance. In: Proceedings of International Conference on Machine Learning 2014 (ICML 2014), p. 754–762 (2014)
11. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
12. Lipp, M., et al.: Platypus: software-based power side-channel attacks on x86. In: 2021 IEEE Symposium on Security and Privacy (SP), pp. 355–371 (2021). https://doi.org/10.1109/SP40001.2021.00063

13. Ltd., P.T.: Picosdk-python-wrappers (2022). https://github.com/picotech/picosdk-python-wrappers
14. McFee, B., et al.: librosa: audio and music signal analysis in python. In: Proceedings of the 14th Python in Science Conference, pp. 18–24 (2015). https://doi.org/10.25080/Majora-7b98e3ed-003
15. Ng, J.S., et al.: A highly efficient power model for correlation power analysis (cpa) of pipelined advanced encryption standard (aes). In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5 (2020). https://doi.org/10.1109/ISCAS45731.2020.9180778
16. Patel, S.N., Robertson, T., Kientz, J.A., Reynolds, M.S., Abowd, G.D.: At the flick of a switch: detecting and classifying unique electrical events on the residential power line (nominated for the best paper award). In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 271–288. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74853-3_16
17. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
18. Slepian, D.: Prolate spheroidal wave functions, Fourier analysis, and uncertainty - V: the discrete case. Bell Syst. Techn. J. **57**(5), 1371–1430 (1978). https://doi.org/10.1002/j.1538-7305.1978.tb02104.x
19. Szefer, J.: Survey of microarchitectural side and covert channels, attacks, and defenses. J. Hardware Syst. Secur. **3**(3), 219–234 (2018). https://doi.org/10.1007/s41635-018-0046-1
20. Virtanen, P., et al.: SciPy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
21. Waskom, M.L.: Seaborn: statistical data visualization. J. Open Source Softw. **6**(60), 3021 (2021). https://doi.org/10.21105/joss.03021
22. Wolf., Z., Larson., E.C., Thornton., M.A.: Data leakage in isolated virtualized enterprise computing systems. In: Proceedings of the 9th International Conference on Information Systems Security and Privacy - ICISSP, pp. 118–123. INSTICC, SciTePress (2023). https://doi.org/10.5220/0011691900003405