Tremors: Privacy-breaching Inference of Computing Tasks using Vibration-based Condition Monitors

Anandarup Mukherjee, *Graduate Student Member, IEEE*, Pallav Kumar Deb, *Graduate Student Member, IEEE*, and Sudip Misra, *Senior Member, IEEE*

***Abstract*—We propose the adaptation of vibration-based condition monitoring systems and techniques, popularly used in industrial condition-based maintenance, for identifying the possibility of compromising the privacy of personal computing systems. This work exploits the automated fan-based heat dissipation features and read/write operations of disk-based storage, commonly present in personal computers, to read computing task-specific vibration signatures on the computer's cabinet/case. These vibration signatures are then used to identify the broad classes of tasks being executed on a separate computer without ever needing to log into the monitored machine. This work builds upon the premise that heterogeneous tasks have distinct computing requirements, which translates to variations in the amount of heat generated by the computer's processor, eventually leading to variations in the computer's heat control fan speed. The variations in the fan's speed and the frequency of read/write operations to disk-based storage create unique vibration signatures, which maps uniquely to the computer's processing operations, leading to a breach of privacy of the computer. Our work's preliminary results suggest that computer-based tasks can be mapped from their vibration signatures with an accuracy of at least 70%. We additionally study the task identification granularity of such an approach.**

***Index Terms*—Condition monitoring, Computers, Security breach, Vibration monitoring, Machine learning, Superlearners, Industrial Internet of Things.**

I. Introduction

Industry 4.0, coupled with the Industrial Internet of Things (IIoT), has brought together machines/devices with high-speed networking, facilitating rapid automation and data transfer on the factory floors. Data-driven techniques for condition-based monitoring and maintenance are gaining popularity and immediate acceptance in diverse industrial domains. Industrial monitoring systems typically rely on a heterogeneous set of sensors for quantifying various physical behaviors of industrial equipment such as vibrations, pressure differences, rotatory motions, and acoustic signatures, generating massive unstructured data and its challenges [1]. The data containing this quantified information from pieces of equipment vary in volume, velocity, and variety, and there exist state-of-the-art methods for handling them efficiently. However, each industrial application generates sensor data signatures, which are uniquely identifiable with each industrial task [2], [3].

A typical IIoT-based industrial monitoring solution comprises the following consolidated functional blocks – sensing, data communication, data storage, analytics, and decision. These functional blocks can be easily created using easily available, cheap, and off-the-shelf sensors and components such as Arduino and Raspberry Pi boards. Recreating these often complex industrial monitoring systems using off-the-shelf components makes these systems highly compact and portable. Although compact, these systems are powerful enough to analyze consumer equipment's mechanical behavior. Some of these consumer equipment may include *personal computers*. In this work, we highlight the possibility of a side-channel attack, which involves identifying the operations on consumer equipment with standard condition monitoring modules.

A. Vulnerability of Traditional PCs

Traditional computing systems such as personal computers (PCs) and servers are made up of a central processing unit (CPU), volatile memory, and disk-based storage (HDD) memory. Although modern computing systems rely on solid-state storage drives (SSD), the presence of disk-based storage is still very common, especially owing to the low cost of such systems. Additionally, irrespective of HDD and SSD, fans for temperature control still exist. The motherboards connecting the various discrete components necessary for the functioning of a full-fledged computer, mostly have automatic heat dissipation mechanisms in the form of electrical motor-based fans. The majority of the heat generated in a PC is directly proportional to the number of computations a CPU has to perform [4]. Relatively recent motherboards have automated mechanisms, that monitor the increase in temperatures inside the housing/cabinet of computing devices, and have the ability to control the speed of these fans. The changes in fan speeds attached to the housing/cabinet of a PC and the disk read-write operations materialize in the form of vibrations, especially of the PC cabinet. These vibrations correlated to PC operations can be used as signatures for specific tasks being performed in the PC. Higher the computations a CPU has to perform or increase in read-write operations to the hard disk results in increased vibrations. The vulnerability of a PC arises when, for example, an accelerometer sensor strategically placed on the PC cabinet is used to detect the changes in the vibration signatures and reverse-engineer the information of the process running in the PC. Such vulnerabilities are beyond implicit security approaches such as in [5].

Example Scenario: An adversary may use the proposed machine learning-based model that identifies application signatures generated in the form of vibration and heat dissipation, for computer programs on their private PC. The adversary may attach (maliciously) a similar set of sensors on the

A. Mukherjee, P. K. Deb, and S. Misra are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. e-mail: anandarupmukherjee@ieee.org, (pallv.deb,sudipm)@iitkgp.ac.in

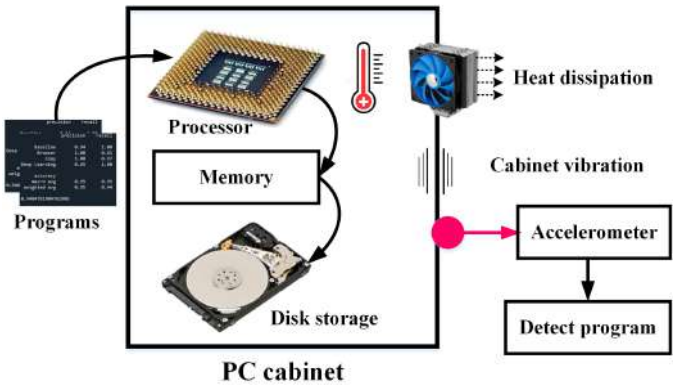


Figure 1: Overview of the proposed system

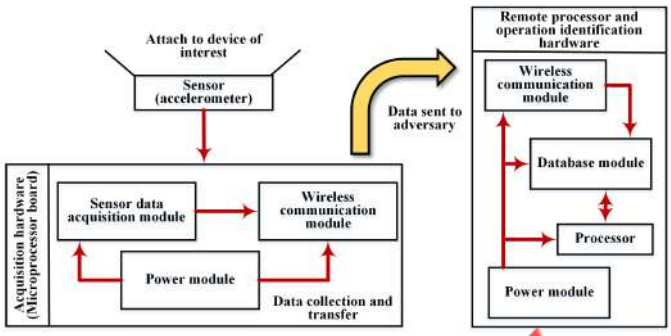


Figure 2: Schematic diagram (hardware components)

computer systems of interest or may use the accelerometer on a smartphone. Without logging into the system, the adversary gains the ability to identify the applications/routines running on the target PC, without ever needing to log into the system, simply based on the trained model from his PC. This scenario certainly poses a threat to the privacy of traditional computing systems, which might necessitate a relook at the way traditional computing systems are designed.

B. Research Questions

In this work, we ask the following research questions in the context of these low form-factor monitoring systems (in our case, an accelerometer-based vibration monitoring system), which functionally mimic industrial monitoring systems:

RQ1: Can we use industrial monitoring solutions for monitoring computing devices?

RQ2: Can industrial monitoring solutions quantify the mechanical and environmental variations caused by the mechanical changes of the supporting hardware of computing devices?

RQ3: Can individual tasks on a computer be identified based merely on these quantified mechanical signatures?

RQ4: Do industrial monitoring systems pose a threat to the privacy of existing computing platforms? If so, to what extent?

C. What we Propose

In this work, we outline and demonstrate the perceived threats to the privacy of traditional computing systems from IIoT-based industrial condition monitoring systems. We propose the use of accelerometer sensors, which are commonly used for monitoring mechanical vibrations in industrial equipment, for detecting the vibrations on the cabinet of a PC. Figures 1 and 2 outline the scientific philosophy and hardware schematic of our approach, respectively. The acquisition unit consists of the sensor (accelerometer) and a wireless communication module (including a power module). The wireless module transfers the data to a remote adversary, who then processes and identifies the running operations. Owing to the foundational nature of this work, as a starting point, we identify three broad classes of PC-based tasks depending on the amount of computation C and frequency of changes in computation Δc. The choice of these three classes allows us to have coarse, but distinctly identifiable vibration signatures on the PC cabinet – Δc_0 (low), Δc_1 (medium), and Δc_2 (high).

The first class (Δc_0), which we also refer to as the 'baseline' in the rest of this paper, consists of regular tasks, not explicitly initiated by the user. Tasks such as basic operating system (OS) functions, software updates, network connectivity, system monitors, and others such for the crux of this class. In regular PCs, from the perspective of a normal user, this is the state of the PC after start-up, when no additional software or programs are being executed by the user. We consider this class as the one having the least amount of computations (relatively) when compared to the other two classes.

Assumption 1. *It is assumed that the PC is not running any malware and is operating within its normal bounds.*

The second class (Δc_1) consists of applications/programs considered to have mid-level computation requirements. These applications are run on top of/ in addition to the baseline operations. Tasks/programs such as browsers, online video streaming applications, word processing software, and others fall in this category. This class of vibration signature mostly relies on the volatile memory (RAM, cache) for their sustained functions. They seldom require high computations and frequent periodic access to HDD storage.

The third and final class (Δc_2) consists of applications and programs, which are similar to Δc_1 run on top of baseline operations. The applications/programs in this class of vibration signatures are characterized by high CPU load in the form of increased processing and frequent access to HDD for storing and retrieving data during the course of their operation. Applications/programs such as deep learning, high-end image editing software, and others fall in this category.

Assumption 2. *Traditional computing systems have an accessible PC cabinet, have automated fan-based heat dissipation mechanisms, and have moving disk-based storage for long-term storage of data.*

D. Contributions

In this work, we raise the question of whether data-driven methods and advancements in industrial monitoring systems and omnipresent structural homogeneity of traditional computing systems pose a threat to the privacy of PCs. In line with

the proposed research questions in Section I-B, we contribute the following through this work:

- We construct a pocket-sized, accelerometer-based vibration measurement system, which can be placed on a PC and is able to wirelessly transmit these signals to a remote system.
- We collect and analyze the vibration signatures of various pre-defined PC-based tasks from a test PC.
- We develop an algorithm for automatically classifying the captured vibration signals into one of the pre-defined classes.
- We use the trained model from the test PC vibration signatures to identify tasks in an entirely new PC and validate the accuracy with which the tasks are identified.

This work focuses on highlighting an overlooked vulnerability concerning the privacy of existing computing systems. This study aims to explore the extent to which this vulnerability can be exploited to start discussing the counter-measures. This work does not raise any ethical concerns. Conventional security challenges in IoT devices and their solutions may be found in [6].

We organize the rest of the paper as follows. We highlight some of the existing literature in Section II. We then describe the time-series information from the PC vibration signatures on running the different applications in Section III, followed by that in the frequency domain in Section IV. We then present the possibility of breaching privacy based on the mentioned signatures in Section V. We explain our setup in Section VI and then present the performance of the developed model in Section VII along with possible countermeasures in Section VIII. Finally, we conclude in Section IX.

II. Related Work

A. Vibration-Based Applications

Vibration data has been used for accomplishing multiple applications. The authors in [7] developed a system consisting of vibration sensors for detecting emergencies on a factory floor or large buildings. They identified these emergencies based on the signatures generated from varying movement patterns. Similarly, Nwakanma *et al.* [8] proposed a similar system using Long Short Term Memory (LSTM) model. Further, Yu *et al.* [9] used a similar setup for localizing personnel on production plants and large buildings using deep learning models. Other applications include: vibration-based communications [10], using vibration sensors on smartphones to counter voice synthesis attacks [11], and others.

B. Attacks and Breaches

We present some of the security threats (software and hardware) in real-world deployments. Software-based attacks cause disruptions in the network flow and execution of regular tasks on-board the devices. Differential fault analysis [12] is one of the common software-based threats, which allows adversaries to gain knowledge of the internal states, without the need for ciphertexts. It induces faults into intermediate blocks, which eventually creates anomalies in the subsequent rounds, exposing the internal details. Keyword guessing attacks [13], as the name suggests, also allows the generation of ciphertexts by malicious users. These attacks allow access to undesired users. Such challenges may be overcome by increasing encryption rounds and adding an authentication layer for legible users. The authors in [14] presented Denial of Service (DoS) attacks, which is a different category of attack and is usually achieved by flooding the network. They proposed using the Tiny Encryption Algorithm (TEA), a block cipher technique, to overcome such issues. However, TEA has its limitations as it uses similar keys. Fault attacks are another form that injects faults into the regular operation routines creating disruptions. The authors in [15] proposed using Advanced Encryption Standards (AES) to overcome such challenges. On other lines, the authors in [20] demonstrated a method of side-channel attack using the acoustic information generated on typing on the keyboard while on a Skype call. Aliyu and Rahulamathavan [21] demonstrated similar attacks on using the soft keyboards on Android smartphones.

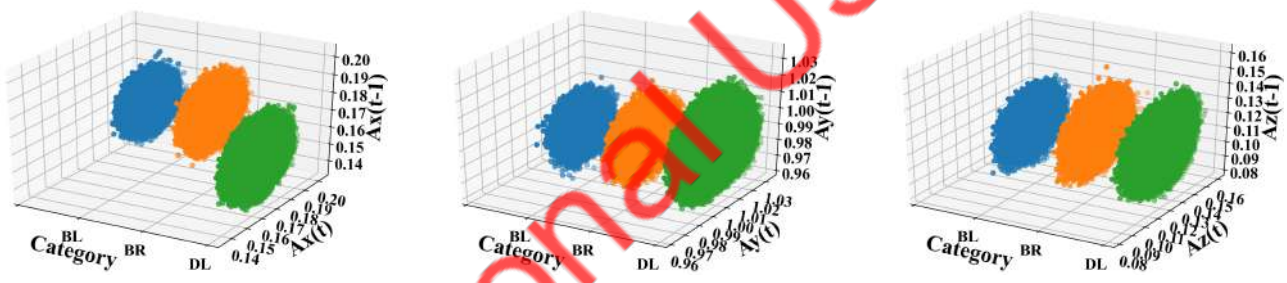
On the other hand, some attackers target the hardware. Mukherjee *et al.* [16] presented Hardware Trojan Attacks and emphasized on its low resource footprint. Such attacks may be mitigated by adopting split manufacturing. However, attackers gain access to the hardware from different manufacturers by inducing modeling attacks [17]. The authors in [19] presented how attackers overwrite memory fragments in buffer overflow attacks, which requires enhanced security hardware designs. Additionally, Dang *et al.* [18] presented Linux-based IoT device attacks. It is a fileless type of attack and is of particular importance as most IoT deployments use Linux-based operating systems. They suggested using non-root users and periodic shell command monitoring to overcome the attack. On the other hand, Mehrabi *et al.* [22] proposed using elliptic curve cryptography (ECC) for enhancing hardware security.

C. Data Forensic and Security

Data security practices in industries are of significant importance to preserve device privacy and to secure operational activities. In this section, we present some of the popular security practices. The authors in [23] highlighted the data vulnerabilities in communication networks. They presented a secure decision-making method by ensuring trust among the devices in the network, such that they do not tamper with the data. Alrahis *et al.* [24] designed key-gate structures to confuse machine learning-based attacks on the hardware. The authors in [25] proposed a secured hardware virtualization-based resource splitting method for overcoming microarchitecture state attacks. Sanchez *et al.* [26] proposed an end-to-end solution for I4.0. They incorporated a Defense in Depth (DiD) strategy coupled with lightweight encryption schemes for securing their network and devices. Deebak and Al-Turjman [27] also proposed a lightweight privacy preservation scheme for IoT networks. They proposed a smartcard-based authentication scheme, satisfying the Authentication and Key Agreement (AKA) properties for securing big data. Comprehensive studies on using fog computing as a security layer ay be found in [28], [29]. On the other hand, the authors in [30] highlighted the security practices and challenges for Supervisory Control and Data Acquisition (SCADA) systems.

Table I: Comparison of attacks and breaches in current literature.

Category	Attack	Characteristic	Remedy
Software-based attacks	Differential fault analysis [12]	Access to ciphertext not requied	Increase the number of computation rounds
	Keyword guessing attack [13]	Any end user may generate keyword ciphertext	Aprior signatures by owners before generating the keyword ciphers
	Denial of Service (DoS) [14]	Achievable through network flooding	Tiny Encryption Algorithm (TEA)
	Fault attacks [15]	Fault injection even in the middle rounds of block ciphers	Advanced Encryption Standard (AES)
Hardware-based attacks	Hardware trojan attack [16]	Low resource footprint	Split manufacturing
	Modelling attack [17]	Threat to physical unclonable functions	Chaotic oscillators
	Linux-based IoT device attacks [18]	Fileless attack	Non root users and auditing shell commands
	Buffer overflow attack [19]	Overwrite memory fragments	Enhanced security hardware design
Sensor-based attacks	Device privacy breach (Proposed work)	Access without credentials	Spread awareness and propose methods



(a) x-axis (Ax) (b) y-axis (Ay) (c) z-axis (Az)

Figure 3: Lag plots of the time-domain accelerometer sensor values for baseline (BL), browser (BR), and deep learning (DL) classes

D. Synthesis

Owing to the condition/tool monitoring methods in this section, we develop the industrial monitoring-inspired pocket-sized vibration measuring unit for collecting data from standard PCs. In such deployments, while both software and hardware-based attacks cause deployment challenges, we summarize how the proposed attack varies in comparison to the others and its remedies in Table I. Irrespective of the solutions for the attacks, the proposed work on the device privacy breach from sensor data is one of a kind. It allows access to the internal working and details of the executing routines without the need for logging into the system. In this work, we attempt to raise awareness about the same and plan to extend this work by proposing solutions for overcoming this issue.

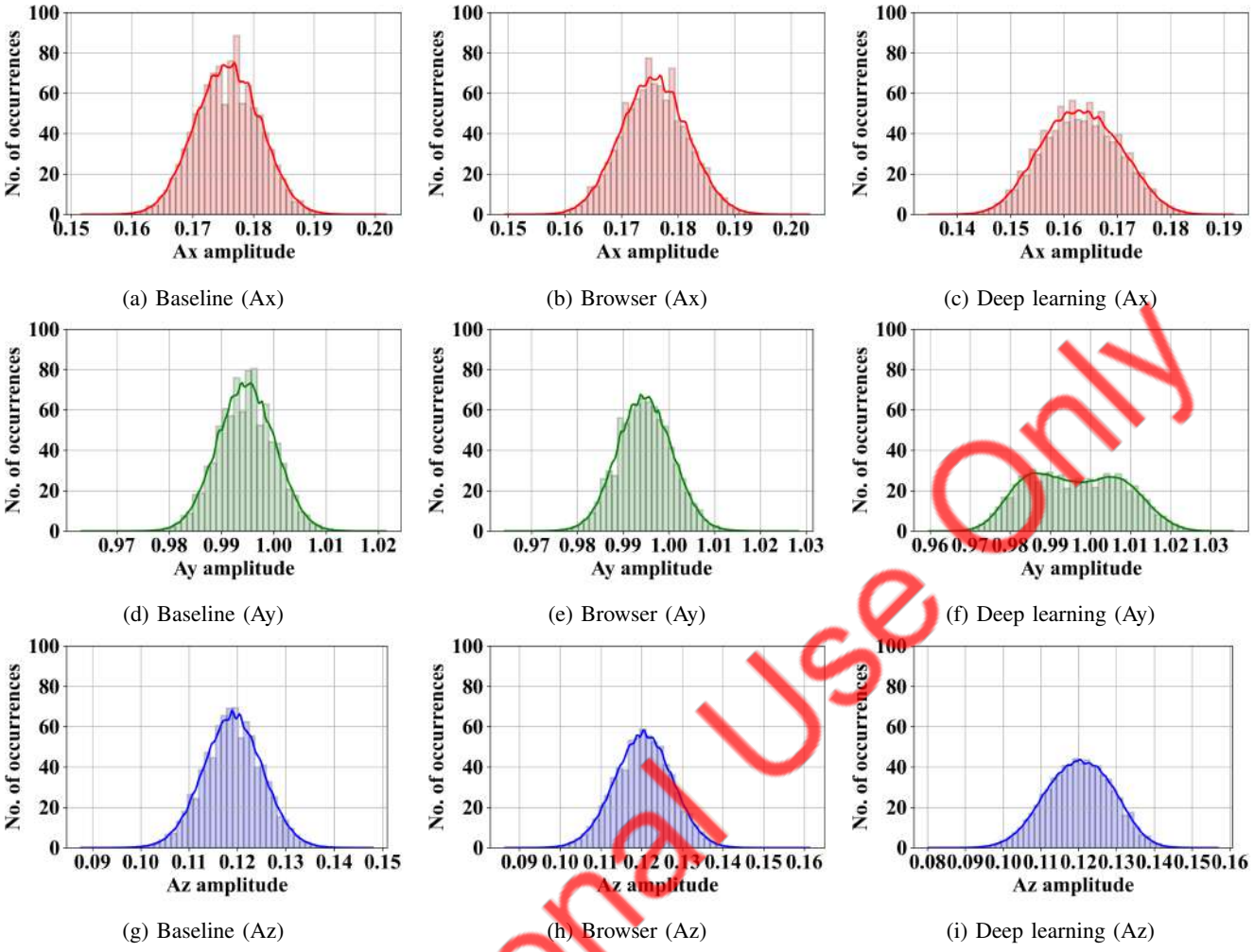
III. Learnings from PC Vibration Signatures

We engineer a system comprising of a digital accelerometer and a WiFi-enabled processor board, which can be easily attached to a PC cabinet. The sensed data from the accelerometer continuously transmits data to a remote server hosting pre-trained classifiers for identifying vibration signatures. This system is functionally analogous to an industrial vibration monitoring system, which relies on mechanical variations of the monitored environment for quantifying the physical behavior of the system under observation. This answers our first research question (**RQ1**) that the core functionalities of an industrial monitoring system can be seamlessly recreated on a scale, which is small enough to be used for monitoring a PC or a consumer computing setup.

In response to the second research question (**RQ2**) on whether the minuscule mechanical vibrations of the supporting PC hardware components corresponding to changes in the PC's changing computing requirements can be distinctly quantifiable, we describe the data collected in the subsequent section (Section III-A).

A. Temporal Data Description

The time-domain representation of the collected vibration signatures for the three broad categories – Δc_0, Δc_1, and Δc_2, corresponding to the test PC's baseline, browser, and deep learning tasks. We analyze the temporal variations in



(a) Baseline (Ax) (b) Browser (Ax) (c) Deep learning (Ax)

(d) Baseline (Ay) (e) Browser (Ay) (f) Deep learning (Ay)

(g) Baseline (Az) (h) Browser (Az) (i) Deep learning (Az)

Figure 4: Distribution of vibration signatures for the three categories of data along the x, y, z axes obtained from the accelerometer sensor

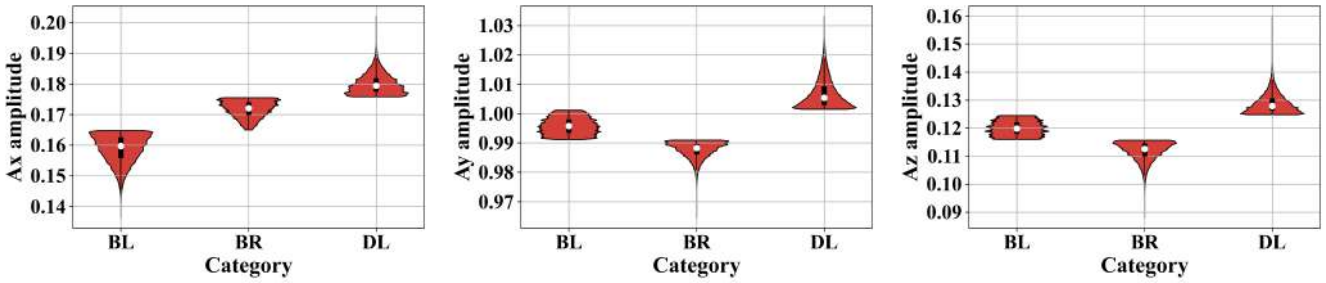
the obtained signal using a lag plot, which is a plot of the variations in the time-domain signal at t (x-axis) versus the variation at $t-1^{th}$ instant (y-axis) for the three categories. Fig. 3 shows the lag plot for the three chosen categories of data, each plot signifying data from each of the accelerometer's axis of observation – Ax (Fig. 3a), Ay (Fig. 3b), and Az (Fig. 3c). Although the lag plot for the first two classes of data – baseline (BL) and browser (BR) – seem identical, the third class – deep learning (DL) – is clearly discernible from the previous two classes. Due to the high density of the data being analyzed, we observe concentrated data points, particularly at the center of each cluster. Interestingly, we observe few or no outliers, which further establishes the stability of the proposed system. Also, each of the clusters in the lag plots follows a circular pattern. This behavior indicates that the data captured by the accelerometer is periodic in nature, further indicating the uniqueness of vibration signatures for each PC-based task/application. Table II represents the standard deviation values of each of the three categories of data for each of the accelerometer's observation axis. The intra-class variations of the standard deviation values in this table further indicates the possibility of having uniquely identifiable task-based signatures in PC, which is in line with **RQ2**.

Table II: Standard deviation of the accelerometer data.

	Baseline	**Browser**	**Deep learning**
Ax	0.0052	0.0058	0.0072
Ay	0.0054	0.0059	0.0118
Az	0.006	0.007	0.008

B. Temporal Data Distributions

To positively confirm **RQ2**, we analyze the temporal vibration signatures using histograms. This helps us to characterize the distribution of each of the task-specific vibration signatures for each of the accelerometer's observation axes. Fig. 4 shows the distribution of vibration signatures for the three categories of data along the x, y, z axes obtained from the accelerometer sensor. We observe that for all cases, the data points roughly follow a normal distribution, but with varying skew and kurtosis. This trend in the distribution of the captured temporal data further leads us to believe that there exist unique and discernible vibration signatures for task-based operations on a



(a) Ax axis. (b) Ay axis. (c) Az axis.

Figure 5: Observations on performing k-means clustering on the axes values on executing baseline (BL), browser (BR), and deep learning (DL) routines

PC, which can be used to map vibration signatures of a PC's cabinet with actual tasks/computations executing within it.

C. Temporal Data Clustering

As a final measure, we cluster the data using *k-means* clustering. We vary the number of clusters (k value) from 1 to 11 and obtain the elbow of the Within Cluster Sum of Squares (WCSS) at the point when k is equal to 3, which is the number of categories in our experiment. Based on this observation, we perform the k-means clustering and present the clusters along with the data points in them in Fig. 5. We observe that the data points, irrespective of the axis, vary distinctively. The Ax axis data points (Fig. 5a) have an increasing pattern for identifying each of the categories of CPU routine. On the other hand, Figs. 5b and 5c show lower Ay and Az values for browser (BR) and baseline (BL) in comparison to deep learning (DL).

This behavior of the captured data, in addition to the ones outlined in Sections III-A and III-B, positively confirm **RQ2**. However, as the inter-category temporal variations are sometimes too small, such as in the case of BL and BR, these trends may not be fully captured while employing an automated data classification mechanism using machine learning.

IV. F-Domain Conversion and Data Shaping

To address the third research question (**RQ3**), we need to adopt a method, which would be – 1) free from temporal artifacts not belonging to the task signatures, 2) able to identify minute differences in the vibration signatures, and 3) the separation between the vibration signatures should be distinct enough to be put in an automated classifier. Towards this, we convert the time-domain vibration signatures to frequency-domain (F-domain) to enhance the vibration signatures by removing any temporal dependencies the signal may have. Temporal dependencies may often lead to unwanted signal artifacts, which are sensor or device-dependent (in turn affecting the clarity of signal segregation of the individual tasks). Initially, we split the temporal signals into segments 1 second wide, which corresponds to the actual sensor sampling rate of $500Hz$. Considering $a_x(t)$ as the original signal, for a sampling rate N, we split the data to generate an array of segmented vibration signatures $A_x(T_i) = [a_x(t_i) \dots a_x(t_{i+1})]$ at any arbitrary time T_i. We choose the value of N in compliance with the Nyquist criteria. For a segmented signal $a_x(t_j)$, we compute its FFT. This operation can be represented using the following expression:

$$a_x(t_j)_{FFT} = \sum_{n=0}^{N-1} a_x(t_j) e^{-\frac{i2\pi kn}{N}} \tag{1}$$

where $e^{-\frac{i2\pi kn}{N}}$ represents the N^{th} roots of unity for $k = 0, 1, \dots N-1$. We repeat this over the entire signal (or the array $A_x(T_i)$) and obtain,

$$\begin{aligned} F[A_x(T_i)] =& [a_x(t_0)_{FFT}\, a_x(t_1)_{FFT}\, \dots\, a_x(t_j)_{FFT}]^T_{j=(\frac{\delta}{N}-1)} \\ =& \begin{bmatrix} \sum_{n=0}^{N-1} a_x(t_0) e^{-i2\pi kn/N} \\ \sum_{n=0}^{N-1} a_x(t_1) e^{-i2\pi kn/N} \\ \dots \\ \sum_{n=0}^{N-1} a_x(t_{\delta/N-1}) e^{-i2\pi kn/N} \end{bmatrix} \end{aligned} \tag{2}$$

where δ is the length of the signal $A_x(T_i)$. We then bin the FFT coefficients of each split segment (10 bins in our case) over the entire frequency range to capture the approximate distribution of frequencies. We bin the signal frequencies and calculate the corresponding values as $B = \sum_{i=0}^{N} c$, such that,

$$c = \begin{cases} 1, & \text{if } FFT_{coeff} > 0 \\ 0, & otherwise \end{cases} \tag{3}$$

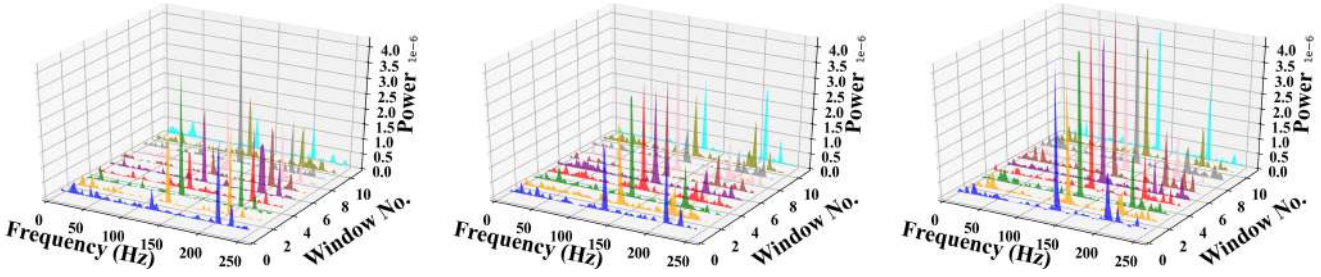
which represents that we count only for those frequencies which have some value for FFT coefficients (FFT_{coeff}).

We repeat these steps for the data corresponding to each axes. On obtaining all the binned data for the three accelerometer axes (Ax, Ay, Az), we calculate the cumulative sum of each frequency range, therefore resulting in a 1D array. This array is the input for our classifier. In summary, we obtain data in the form:

$$B_{i,j} = \begin{bmatrix} B_{1,1} & B_{1,2} & \dots & B_{1,i} \\ \dots & & & \\ B_{j,1} & B_{j,2} & \dots & B_{j,i} \end{bmatrix} \tag{4}$$

where $i \in [1, 10]$ and $j \in [1, \delta]$.

To demonstrate that splitting the full-range time-domain signature of the activities does not hamper the nature or the information content of the signal, we consider the total entropy of the signal and the entropy of the signals after it was split. For the FFTs (m^{th} window), we calculate the power spectrum as $S(m) = (a_x(t_m)_{FFT})^2$ and the probability density as



(a) Baseline. (b) Browser. (c) Deep Learning.

Figure 6: PSD for the segmented signals (1 second wide) for all three vibration signature categories

$P(m) = S(m)/\sum_i S(i)$. The normalized spectral entropy is then calculated as:

$$H_n = -\sum_{m=1}^{N} \frac{P(m) log_2 P(m)}{log_2 N} \quad (5)$$

As H_n relies on the power spectrum $S(m)$ of a signal, we plot the variations in the Power Spectral Densities (PSD) of the three data categories after (refer to Fig. 7) and before splitting the signal (refer to Fig. 6). For the purpose of demonstrating the rationale behind splitting the signal, we only illustrate the plots using the signatures for the accelerometer x-axis. In Fig. 6, we observe that the PSD differs as the vibration signature data category changes. In the case of baseline (Fig. 6a), we observe the maximum spectral strength is $2.5 \times 1e-6$ units. Similarly, we observe a similar maximum spectral strength in the case of browsing (Fig. 6b), but with higher densities spread across the spectrum bands. In the case of deep learning (Fig. 6c), we observe the maximum entropy, in the range of $4 \times 1e-6$ units. We attribute these observations to the variation of the speed of the cooling fan, which proportionately increases or decreases the vibrations of the PC cabinet. It may be noted that PSDs in Fig. 6 follow a similar trend as its neighbors, proving that the information content remains relatively unchanged for the segments even on splitting.

Further, we plot the PSDs of the entire range of the signal (without splitting) for the three vibration signature categories in Fig. 7. We observe that the PSDs in this case follow a similar trend to the PSD in the case of the split/fragmented signals. Interestingly, the difference between the entropies under each category of execution routine remains the same, proving that the splitting of the signals preserves the signal characteristics and also demonstrates the stability of our proposed approach.

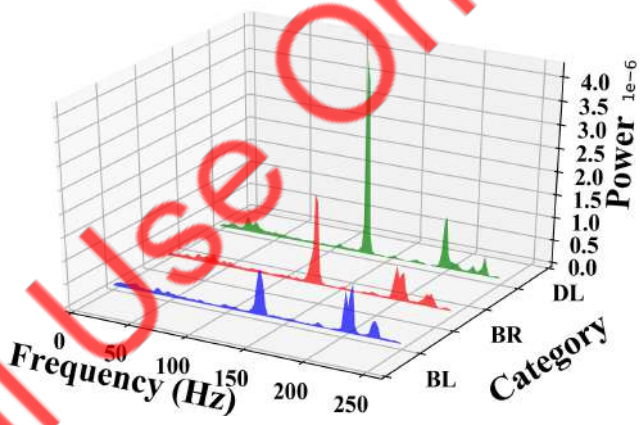


Figure 7: PSD of the entire signal for the three vibration signature categories

V. Vibration-based Privacy Breaching

A positive response for **RQ3** leads us to further investigate if the uniqueness of computing task-based vibration signatures can be used to breach the privacy of computing systems. For this, we develop a methodology, which relies on identifying the frequency-domain information from the collected task-specific vibration signatures. This frequency-domain information is used to train a super learner. Fig. 8 depicts the proposed methodology for developing the system that identifies the category of routines executing on a computer. We obtain the accelerometer data (Ax, Ay, and Az) from the externally positioned accelerometer sensor unit on a standard PC cabinet. This time-domain data from each of the accelerometer axes is split into windowed samples of 1 second each, which corresponds to the sensor sampling frequency of 500Hz. This makes the number of data/sample points equal to the sampling rate of the sensor. Each of these split time-domain samples is converted to the frequency domain using FFT. Subsequently, a binning operation (refer to Section IV and Equation 3) bins the coefficients of the FFT operation into 10 bins. We add the signals from the three channels (accelerometer x, y, and z axes) to capture any/all variations in the vibration signatures. This step also has the added advantage of making our approach independent of the orientation of the accelerometer sensor and enhancing the repetitive frequency components present in all three axes of the accelerometer data. The data in this form is analogous to that of Equation 4.

We train multiple classifiers on this data to determine the one which offers superior results. Fig. 9 shows the performance of the classifiers trained against the three data categories. We observe that these classifiers tend to confuse some of the vibration signatures in a class with that of other classes. For instance, all the classifiers efficiently identify the deep learning task (bottom right cell of each confusion matrix). However, most of the classifiers consistently predict erroneously some instances of the tasks for the baseline and browser category of data. This effect becomes prominent for

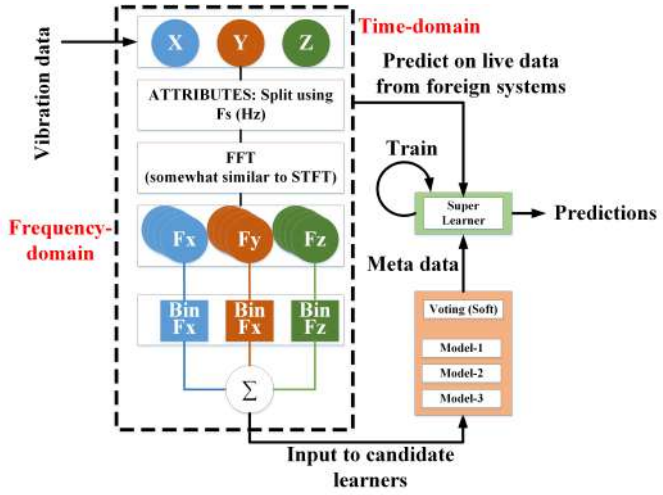


Figure 8: Activity identification methodology

unseen test data. This is because the two routines mostly involve similar OS operations, apart from some cache usage, HTTP requests, and the corresponding reply reception. Additionally, we also attribute this behavior to the working principle of the classifiers. For instance, Random forest, Extra Trees, and AdaBoost focus on decisions from multiple decision trees and then generalizing the output. The splits of the decision trees are based on the purity of the data and operations such as BL and BR (which are minutely different) tend to represent similar features. Further, the varied outcome in AdaBoost is intuitively because of the boosting method. It diminishes the possibility of classifying BL. The KNN model also reports similar results and we attribute it to a similar reason that the data points for BL and BR lie close to one another. Gaussian Naive Bayes and SVC demonstrate better classifications as they operate on a combination of probabilistic and statistical methods, which has a better insight into the difference between the two activities. Irrespective of the listed flaws, each of these models has its own set of unique features, which we tend to exploit to get a better insight into this work. Consequently, we make use of a super-learner [31], in conjunction with the classifiers to enhance the classification accuracy of our approach. A super learner involves performing a k-fold split training and evaluation on a list of machine learning based models that individually perform well on a given dataset. After the cross-validation of the model, the super learner uses these out-of-fold predictions along with the entire dataset for training a unique/final model. It performs the same or better than the best individual classifier model, guaranteeing no under-performance.

VI. Experiment Design

We use a desktop computer for obtaining the initial vibration data from the three task categories for demonstration purposes. This data acts as the training data for our proposed approach. Subsequently, we use a laptop to obtain its vibration data and categorize its tasks into the three task groups through our trained classifier. We use an MPU6050 accelerometer sensor interfaced using a NodeMCU microprocessor. The data from

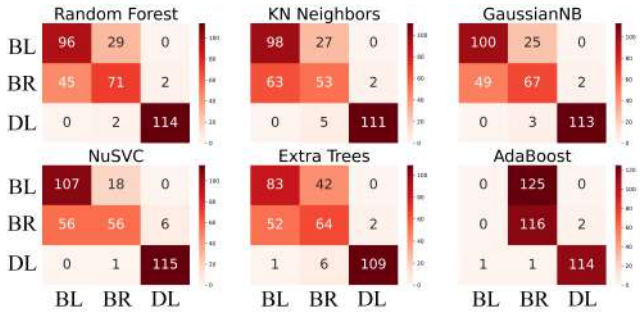


Figure 9: Confusion matrices from super learner base models

the IMU-attached microprocessor is collected in a separate computer. We attach the sensor arrangement on the surface of both the computers for recording the vibration signatures on executing the categorical tasks mentioned in Section I-C. This experiment's main aim is to evaluate the effectiveness of our proposed approach to use vibration-based signatures for identifying PC-based tasks and highlighting that although the computing platforms with the training and testing data are vastly different, it still identifies task categories successfully.

A. Data Collection

The data collection mechanism consists of an Inertial Measurement Unit (IMU) sensor attached to a low-cost Wi-Fi enabled microprocessor. We first calibrate the IMU (MPU6050) with a sampling rate of 500 Hz. As mentioned earlier, we use two data sources – a desktop PC and a laptop. The standard Desktop PC (PC1) has an i3 core processor and 16GB RAM, with disk-based storage of 500 GB. The computing components of PC1 are housed in a tower cabinet, with provision for motherboard-triggered control of cooling fan speeds in response to changes in the cabinet temperature. We use vibration data from this computer for training the super learner using our proposed approach.

On the other hand, we consider an i5 core processor Dell Inspiron laptop (PC2) with 8 GB RAM and 1 TB disk-based storage for validating our approach. Although the characteristics of the components on-board both PCs are the same, they have different form factors. Additionally, laptops generate lesser vibrations than cabinet-based computers due to the compact arrangement of components inside the laptop. We use these setups for demonstrating the working of our proposed approach based on the following categorically identified tasks.

B. Categorical Tasks

We set our Baseline (BL) when the PC is in its idle state and consider this as a Δc_0 category task. We terminate all (unnecessary) background processes while capturing this class's data using the IMU sensor. We record a combination of baseline category data with and without Internet connectivity. For Δc_1 tasks, we consider activities related to web browsing. In particular, we play videos on streaming sites such as YouTube and Netflix. Lastly, we consider training various deep learning models like the Δc_2 task as it consists of high computations and repeated access to the disk-based storage

Table III: Metrics on training the proposed super learner with AdaBoost (AdB), Decision Tree (DT), Extra Trees (ET), Gaussian Naive Bayes (GNB), K-Nearest Neighbors (KNN), Support Vector Classifier (nuSVC), and Random Forest (RF).

Models (atomic)	Score		Fit time (s)		Test time (s)	
	Mean	Std.	Mean	Std.	Mean	Std.
AdB	0.64	0.03	51.31	0.14	0.84	0.08
DT	0.69	0.03	3.64	1.24	0.33	0.36
ET	0.69	0.03	199.47	0.73	27.37	0.52
GNB	0.76	0.04	2.34	1.11	0.72	0.37
KNN	0.72	0.03	0.42	0.44	22.19	1.77
NuSVC	0.65	0.08	0.77	0.22	0.28	0.17
RF	0.76	0.04	110.39	2.38	5.06	0.56

for read/write tasks while training the model consisting of a series of feed-forward and backpropagation operations. We used Long-Short Term Memory (LSTM) for training time-series data and a Transfer-learning model for training images under the deep learning category of tasks. The selection of these deep learning tasks ensures storage of the weights and repetitive input and output operations from both primary and secondary memories. It may be noted that while executing the tasks on the computer systems, we terminated all background processes attributed to non-essential OS operations of these computers. This was done to ensure that the data collected was with minimal vibration signature influence from other tasks. For instance, we disconnect from the Internet to avoid unauthorized downloads and updates and any other operation that may affect the PC components' vibration signatures. We collect the following data from PC1 and PC2:

1) *Training data from PC1 (desktop)*: We collect a minimum of 15 minute vibration trace of each of the categorical tasks (baseline, browser, deep learning). This data is used for training the super learner.
2) *Validation data from PC1 (desktop)*: We collect smaller $2-3$ minute vibration traces for specific tasks falling in one of the three identified categories. Some of these are baseline, Youtube (browser), Netflix (browser), Video conferencing over Microsoft Teams (browser), Python-based LSTM model (deep learning), Python-based Transfer learning (deep learning).
3) *Validation data from PC2 (laptop)*: We collect smaller $2-3$ minute vibration traces for specific tasks falling in one of the three identified categories. Some of these are baseline, Youtube (browser), Netflix (browser), Python-based LSTM model (deep learning), Python-based CNN (deep learning).

VII. Performance Evaluation

A. Training the Super Learner

We automate the detection of task categories based on vibration signatures by training a super learner along with the base models enumerated in Table III. We choose these models as they provide superior performance in terms of post-training metrics, as shown in Fig. 9. On training each of the models, we obtain an average accuracy of 70%. These models contribute

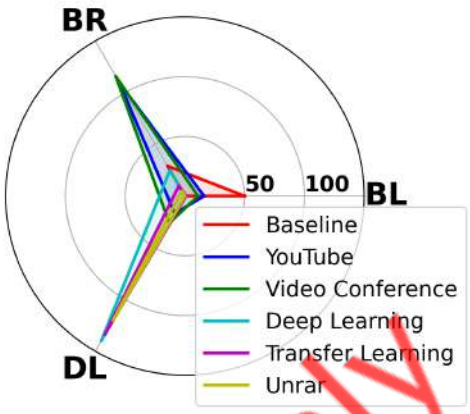
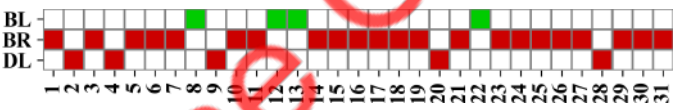
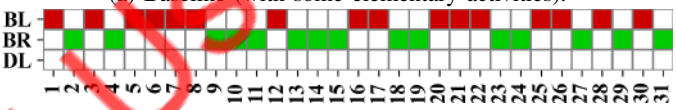


Figure 10: Correct predictions instances of the validation data from PC1



(a) Baseline (with some elementary activities).



(b) Netflix.

Figure 11: Uncertain predictions on validation data from PC1 (green for correctly classified and red for incorrect ones)

to the proposed super leaner's training, which finally reports an accuracy of 79% on the k-fold training data. As shown in Table III, the super learner takes significant time for training as forest-based models such as Extra trees and Random Forest take almost 200 seconds for training. However, they take less test time compared to the k-nearest neighbor method.

We evaluate the super learner trained in Section VII-A on data from the same standard computer we used to obtain the training dataset. We input the consolidated binned data of the three axes and record the predictions. We observe that the model identifies the baseline accurately on a majority of the samples (Fig. 10). On the other hand, it classifies mouse clicks and rollovers (baseline in Fig. 11a) into the browser category, indicating an activity that is separate from baseline tasks and correctly identifies the presence of an active user. In the case of Netflix (Fig. 11b), the model classifies it as baseline on some occasions in contrast to that of YouTube. The model accurately identifies browser activity in the case of YouTube in almost all instances. We attribute this difference in behavior to the way these two streaming services buffer their data in memory. The super learner classifies memory and processing-intensive tasks in the deep learning category (Fig. 10). This behavior is expected for tasks needing to undergo above-median computations and repeated read/write access to RAM and disk-based storage. A detailed explanation of the results from the validation of PC1 and PC2 data is provided in the subsequent section.

B. Data Validation

As mentioned earlier, we validate the developed super learner classifier using the vibration data from both PC1 and PC2 on performing different tasks. We perform multiple tasks on both the computers, which are operationally analogous to the categorical tasks mentioned in Section VI-A. Although we observe promising results during our experiments, we also notice uncertainties in some cases.

1) Predictions on PC1 Activities: Fig. 10 shows the count of correct task-specific predictions made using our proposed approach. The method correctly identifies the PC1's baseline validation data. This is because, during an idle state, the PC is unlikely to make significant changes in its fan and HDD head movements. On performing browser-related activities such as streaming YouTube videos and hosting video conferences, the PC typically uses its cache and some processing for updating the interface screens. The classifier correctly identifies these activities with minor misclassifications in some of the samples of each task data. Interestingly, both behaviors are the same. We attribute this behavior to the conditions that prolonged browser usage generates heat and reduces work on other processes (as the cache gets overloaded). Additionally, we also observe correct predictions on deep learning activities due to the constant high amounts of computations and read/write operations on the HDD. The correct prediction of the un-rar/unzipping compressed files also confirms the uniqueness in vibration signatures of the PC on continuous interaction with the HDD.

However, we observe some dubiety in the classifier predictions (Fig. 11). We run our mouse around the screen and make a few clicks while recording the vibration data for fashioning a variant of baseline. We observe in Fig. 11a, the classifier identifies this as BR in most cases and some as DL. Intuitively, we attribute this to the cache and interface usage processes instead of the earlier baseline activity. We infer that the classifier correctly classifies activity (irrespective of intensity) compared to the idle state, which helps identify if the user is using the PC. Moreover, in addition to watching videos on YouTube, we performed the same activity on streaming videos over Netflix. We observe that the classifiers have mixed predictions on BL and BR (Fig. 11b). This misclassification in the Netflix data is attributed to the way the Netflix platform caches and buffers videos for streaming. Interestingly, since Netflix does not invoke the HDD-based operations, the classifier does not classify it as DL.

2) Predictions on PC2 Activities: We perform similar operations on the laptop and identify tasks using our proposed approach. The laptop being a compact computing device tends to produce much lesser vibrations compared to desktop computers. Similar to PC1, we observe in Fig. 12 that it correctly identifies PC2 in its idle state (baseline). However, in contrast to Netflix in PC1, it correctly categorizes the activity as BR. Moreover, we observe some inconsistency in the case of YouTube (Fig. 13a). Beyond the inconsistency, we notice that the classifier correctly captures the state when the user uses the PC2, irrespective of its form factor and relative stability compared to the tower cabinet in PC1. The laptop's reduced vibrations also cause sub-optimal performance in identifying

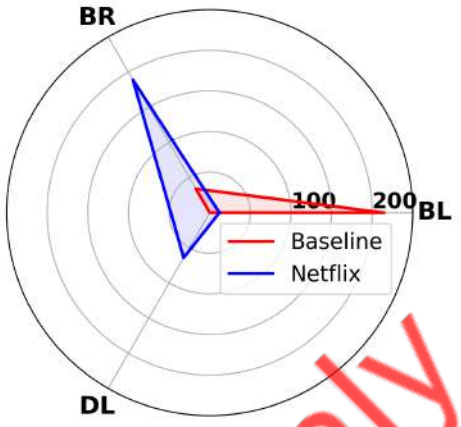


Figure 12: Correct prediction instances of the validation data from PC2

(a) YouTube.

(b) Deep learning.

Figure 13: Uncertain predictions on validation data from PC2 (green for correctly classified and red for incorrect ones)

the deep learning task (Fig. 13b). In comparison to YouTube, as deep learning activity uses more processing and HDD head movement, we notice more BR classifications than BL, which suggests that the sensor and classifier correctly capture the variations in the hardware vibrations. We infer that depending on the PCs' hardware setup, the developed model needs tweaking, which we plan to address in our extended work.

VIII. Discussion and Countermeasures

Through the performance evaluation of our approach, we reiterate that principally, it is possible to train a classifier to read computing task-based vibration signatures from a computer and identify similar task-based patterns in other computers without the need to log-in to the system under observation. This also affirms the initial research questions in **RQ3** and **RQ4**. This approach highlights a significant privacy vulnerability in the existing computing systems, as traditionally, security and privacy approaches were either focused on securing network-based intrusions or required physical access to a computer's internal hardware.

To limit a computing system from privacy breaches using strategies similar to – Tremors, some straightforward and easy-to-implement countermeasures may be adopted. The first and foremost is limiting the cooling fan speed or choosing alternate processor cooling mechanisms such as liquid-based CPU coolers. Secondary approaches include using SSDs instead of disk-based ones to remove any or all vibrations due to

data read/write onto these drives. Finally, some complicated methods may include using an underlying software or code to obfuscate the vibration signatures specific to actual tasks being executed on the computer. However, this approach is not efficient in terms of energy consumption and longevity.

IX. Conclusion

In this work, we demonstrated the possibility of side-channel attacks using off-the-shelf condition monitoring IIoT devices. As proof of concept, we used an accelerometer sensor to read the signatures generated from the cooling fans on a standard personal computer and identify the operations. As an example, an adversary may simply put his phone on top of a PC cabinet and capture the vibration signatures. Through this work, we explored the extent to which the privacy of existing computing systems can be exploited using machine condition monitoring systems, typically employed in Industrial IoT-based monitoring of machinery. We developed and demonstrated how a classifier trained on the vibration signatures of computing tasks based on one computer could be used to identify tasks in a different computer, which the system has not previously seen. Finally, we briefly outline the possible causes of under-performance of our approach and possible approaches to safeguarding a computer from Tremors.

In the future, we aim to refine this approach to enable a more satisfactory detection of individual tasks on a computer, and not just task categories. We also plan to study the uniqueness of vibration signatures generated by the computer's OS-specific operations, to uniquely identify the patterns each software adopts during its operation on a computer. We also plan to study the behavior of the hard disks individually by placing the accelerometer sensor directly on them, rather than only on the cabinet. This will potentially help in generating further granular insights.

References

[1] H. Jo, J. Kim, P. Porras, V. Yegneswaran, and S. Shin, "GapFinder: Finding Inconsistency of Security Information From Unstructured Text," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 86–99, 2021.

[2] G. Sharma, V. Tripathi, and A. Srivastava, "Recent Trends in Big Data Ingestion Tools: A Study," in *Research in Intelligent and Computing in Engineering*. Springer, 2021, pp. 873–881.

[3] K. Basu, S. S. Hussain, U. Gupta, and R. Karri, "COPPTCHA: COPPA Tracking by Checking Hardware-Level Activity," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3213–3226, 2020.

[4] "HP Notebook PCs - Why Computers Generate Heat," accessed on 23 Sep. 2021. [Online]. Available: https://support.hp.com/in-en/document/c02655320

[5] D. Meng, R. Hou, G. Shi, B. Tu, A. Yu, Z. Zhu, X. Jia, Y. Wen, and Y. Yang, "Built-in Security Computer: Deploying Security-First Architecture Using Active Security Processor," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1571–1583, 2020.

[6] P. I. R. Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the Internet of Things: Challenges, Threats and Solutions," *Internet of Things*, vol. 5, pp. 41–70, 2019.

[7] Y. Yu and T. Weis, "A Privacy-Protecting Indoor Emergency Monitoring System Based on Floor Vibration," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, 2020, p. 164167.

[8] C. I. Nwakanma, F. B. Islam, M. P. Maharani, D.-S. Kim, and J.-M. Lee, "IoT-Based Vibration Sensor Data Collection and Emergency Detection Classification using Long Short Term Memory (LSTM)," in *Proceedings of International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2021, pp. 273–278.

[9] Y. Yu, M. Waltereit, V. Matkovic, W. Hou, and T. Weis, "Deep Learning-Based Vibration Signal Personnel Positioning System," *IEEE Access*, vol. 8, pp. 226 108–226 118, 2020.

[10] G. Zhao, B. Du, Y. Shen, Z. Lao, L. Cui, and H. Wen, "LeaD: Learn to Decode Vibration-Based Communication for Intelligent Internet of Things," *ACM Transactions on Sensor Networks*, vol. 17, no. 3, 2021.

[11] S. A. Anand, J. Liu, C. Wang, M. Shirvanian, N. Saxena, and Y. Chen, "EchoVib: Exploring Voice Authentication via Unique Non-Linear Vibrations of Short Replayed Speech," ser. ASIA CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 6781.

[12] N. Vafaei, S. Saha, N. Bagheri, and D. Mukhopadhyay, "Fault Attack on SKINNY Cipher," *Journal of Hardware and Systems Security*, vol. 4, no. 4, pp. 277–296, 2020.

[13] J. Li, M. Wang, Y. Lu, Y. Zhang, and H. Wang, "ABKS-SKGA: Attribute-Based Keyword Search Secure Against Keyword Guessing Attack," *Computer Standards & Interfaces*, vol. 74, p. 103471, 2021.

[14] V. Sharma and A. Sharma, "IoT Security Architecture with TEA for DoS Attacks Prevention," in *Advances in Information Communication Technology and Computing*. Springer, 2021, pp. 215–226.

[15] S. Saha, A. Bag, D. B. Roy, S. Patranabis, and D. Mukhopadhyay, "Fault Template Attacks on Block Ciphers Exploiting Fault Propagation," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2020, pp. 612–643.

[16] R. Mukherjee, V. Govindan, S. Koteshwara, A. Das, K. K. Parhi, and R. S. Chakraborty, "Probabilistic Hardware Trojan Attacks on Multiple Layers of Reconfigurable Network Infrastructure," *Journal of Hardware and Systems Security*, vol. 4, no. 4, pp. 343–360, 2020.

[17] V. S. Balijabudda, D. Thapar, P. Santikellur, R. S. Chakraborty, and I. Chakrabarti, "Design of a Chaotic Oscillator based Model Building Attack Resistant Arbiter PUF," in *Proceedings of the Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 2019, pp. 1–6.

[18] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. A. Chen, T. Xu, Y. Chen, and J. Yang, "Understanding Fileless Attacks on Linux-Based IoT Devices with Honeycloud," in *Proceedings of the* 17^{th} *Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 482–493.

[19] B. Xu, W. Wang, Q. Hao, Z. Zhang, P. Du, T. Xia, H. Li, and X. Wang, "A Security Design for the Detecting of Buffer Overflow Attacks in IoT Device," *IEEE Access*, vol. 6, pp. 72 862–72 869, 2018.

[20] A. Compagno, M. Conti, D. Lain, and G. Tsudik, "Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP," 2017.

[21] H. T. Aliyu and Y. Rahulamathavan, "Type and leak your ethnicity on smartphones," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 2557–2561.

[22] M. A. Mehrabi, C. Doche, and A. Jolfaei, "Elliptic Curve Cryptography Point Multiplication Core for Hardware Security Module," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1707–1718, 2020.

[23] G. Rathee, S. Garg, G. Kaddoum, and B. J. Choi, "A Decision-Making Model for Securing IoT Devices in Smart Industries," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[24] L. Alrahis, S. Patnaik, J. Knechtel, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "UNSAIL: Thwarting Oracle-Less Machine Learning Attacks on Logic Locking," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2508–2523, 2021.

[25] H. Omar, B. D'Agostino, and O. Khan, "OPTIMUS: A Security-Centric Dynamic Hardware Partitioning Scheme for Processors that Prevent Microarchitecture State Attacks," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1558–1570, 2020.

[26] A. Mosteiro-Sanchez, M. Barcelo, J. Astorga, and A. Urbieta, "Securing IIoT using Defence-in-Depth: Towards an End-to-End secure Industry 4.0," *Journal of Manufacturing Systems*, vol. 57, pp. 367–378, 2020.

[27] B. D. Deebak and F. AL-Turjman, "Lightweight Authentication for IoT/Cloud-Based Forensics in Intelligent Data Computing," *Future Generation Computer Systems*, vol. 116, pp. 406–425, 2020.

[28] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "Towards a Systematic Survey of Industrial IoT Security Requirements: Research Method and Quantitative Analysis," in *Proceedings of the Workshop on Fog Computing and the IoT*, ser. IoT-Fog '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 5663.

[29] ——, "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[30] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and

Tactics," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1942–1976, 2020.

[31] M. J. van der Laan, E. C. P. Eric, and A. E. Hubbard, "Super learner," *Statistical applications in genetics and molecular biology*, vol. 6, no. 1, 2007.

Anandarup Mukherjee is currently a Senior Research Fellow and Ph.D. Scholar in Engineering at the Department of Computer Science and Engineering at Indian Institute of Technology, Kharagpur. He finished his M.Tech and B.Tech from West Bengal University of Technology in the years 2012 and 2010, respectively. His research interests include, but are not limited to, networked robots, unmanned aerial vehicle swarms, Internet of Things, Industry 4.0, 6G and THz Networks, and enabling deep learning for these platforms for controls and communications. His detailed profile can be accessed at http://www.anandarup.in

Pallav Kr. Deb is a Ph.D. Research Scholar in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. He received his M.Tech degree in Information Technology from Tezpur University, India in 2017. Prior to that, he has completed the B. Tech degree in Computer Science from the Gauhati University, India in 2014. The current research interests of Mr. Deb include UAV swarms, THz Communications, Internet of Things, Cloud Computing, Fog Computing, and Wireless Body Area Networks. Further details are available in https://pallvdeb.github.io/

Sudip Misra (M09SM11) is a Professor with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur. He received his Ph.D. degree in Computer Science from Carleton University, in Ottawa, Canada, and the masters and bachelor's degrees, respectively, from the University of New Brunswick, Fredericton, Canada, and the Indian Institute of Technology, Kharagpur, India. Dr. Misra is the Associate Editor of the IEEE Transactions Mobile Computing and IEEE Systems Journal, IEEE Transactions on Sustainable Computing, IEEE Network, and Editor of the IEEE Transactions on Vehicular Computing. His current research interests include algorithm design for emerging communication networks and Internet of Things. Further details about him are available at http://cse.iitkgp.ac.in/ smisra/.