ViViSnoop: Someone is Snooping Your Typing Without Seeing It!

Kun Jin[†‡], Si Fang[‡], Chunyi Peng[†‡], Zhiyang Teng[§], Xufei Mao[*], Lan Zhang[⋆], Xiangyang Li[⋆]
[†]Purdue University [‡]The Ohio State University
[§]Singapore University of Technology and Design [*]Tsinghua University [⋆]University of Science and Technology, China

Abstract*—In the paper, we present *ViViSnoop*, a novel video-assisted keystroke inference attack which snoops the victim user's typed input without visually seeing it. Instead, it infers the typed input from the vibration extracted from the video capturing the desk where the physical or virtual keyboard is placed. *ViViSnoop* is built on the fact that keystrokes on the desk incur mechanical vibrations which are subtle but still extractable through computer vision processing. To further exploit subtle vibration patterns for accurate and reliable keystroke inference, *ViViSnoop* incorporates a suite of techniques such as vibration-specific video processing to enhance raw vibration quality, a novel virtual sensing array technique to develop fine-grained location signatures, and a two-phase classifier to achieve high accuracy and efficiency. Our extensive evaluation shows that *ViViSnoop* realizes high-accuracy keystroke inference. *ViViSnoop* achieves around 55% single-character accuracy in inferring passwords (random inputs). In word and sentence inference, it achieves 71.4% and 59.4% accuracy using top-1 choices and even almost 100% and 75% using top-10 choices. This means there is almost no difficulty in understanding user inputs. It imposes a covert and serious threat to leak user typing in public/office spaces.

I. Introduction

Snooping a user's typed inputs, also referred to as a keystroke inference attack, is a serious threat of leaking information and user privacy. With such snooping, it becomes easy for the adversary not only to sneak into sensitive information such as usernames, passwords, credit card numbers, SSNs and confidential documents, but also to retrieve user activities and incur unintentional privacy risks.

In recent years, inferring user inputs from contact-free sensing has been actively studied as an effective means to keystroke inference attacks, especially by leveraging ubiquitous cameras. Most video-assisted studies (except [1]) assume that the adversary is able to video-record the victim user's input process and thus eavesdrop on keyboard inputs from the reflections of the screen [2]–[6], or retrieve the taps on touch-screens via hand movement [7] and finger positions [8], [9]. However, these attacks require that the camcorder capture the user input process with little or no visual obstructions and thus greatly limit their applicability. Sun *et al.* break this requirement in VISIBLE by recording and analyzing the video of the tablet backside [1]. Through analysis of different motion patterns incurred by touches at different positions, it empowers a covert keystroke inference for touch screens without direct views.

Our work is highly inspired by [1], but targets at a generic

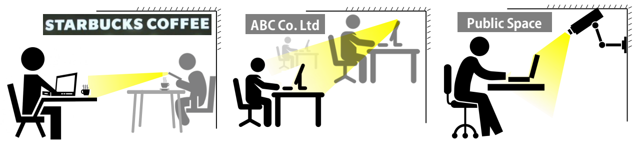


Fig. 1: Typical attack scenarios at the cafe (public space, left), open office (with multiple PCs, middle), and other public spaces with surveillance cameras (right).

keyboard emanation. It works not only for both physical keyboards used by laptops and desktops, but also virtual keyboards used by tablets and phones. Figure 1 illustrates three typical attack scenarios, in which the victim user types her/his input on a table and the nearby camcorder is able to record the surrounding of the victim user without visually seeing the typing behaviors behind the obstacle such as a laptop lid, a computer screen and human bodies. Instead of retrieving *direct* motion patterns on the backside video [1], we exploit *indirect* vibrations occurred on the table, due to keystrokes at different positions. We notice that the idea of detecting vibrations for keyboard emanation has been explored in [10] but they use the victim's smartphone to record the accelerometer data when it is placed very close to the keyboard (2 cm apart). However, our attack is much more realistic and threatening as it does not require any malware or compromise on the victim's side. It is also much less noticeable through a contact-less recording from a distance.

In particular, we propose *ViViSnoop*, which snoops the typing inputs through **Vi**brations extracted from the **Vi**deo captured at a distance. It uses a built-in commodity camcorder like a smartphone camera, a PC webcam, and even a surveillance camera, *etc.*. It records seemingly *irrelevant* videos capturing the computer desk surface surrounding the keyboard, with no need to videotape hand movement, fingertips or anything directly relevant to user inputs like [2]–[9]. *ViViSnoop* incorporates video processing and machine learning techniques to extract subtle vibrations and eventually infer the typed inputs.

To achieve this, we have to overcome three main challenges. First, the vibrations incurred by user keystroke are very subtle when extracted from the video captured at a distance. There are many noise sources such as people walking, hand movements, recording distance and angle, and the camera sensor physical limitations. All these require effective methods to detect and extract high quality vibrations. To this end, we enhance vibration extraction through vertical projection and noise filtering based on mechanical characteristics, Second, the vibration patterns caused by distinct keystrokes are very close and hard to distinguish. To retrieve fine-grained (cm-level) keystroke signatures out of subtle vibrations, we propose a novel virtual sensing technique that retrieves multiple vibration signals caused by the same keystroke but observed at different desk positions (aka virtual sensors). We further exploit the delta among different observations to develop a more "directional" feature vector. Last but not least, individual keystroke detection is still not very accurate, especially in a small training set. It further limits the accuracy of inferring text inputs. We address this issue through a hierarchical classification in which we roughly classify the close keys into clusters at the coarse-grained phase and then exploit the linguistic relationship in order to improve the final inference accuracy.

We have conducted extensive experiments to evaluate *ViViSnoop* using three physical keyboards and a virtual keyboard on an iPad mini 1. Our experiment results show that *ViViSnoop* poses a serious threat in public/office spaces. It infers a single keystroke with an average accuracy of 50%, which outperforms the most relevant state-of-the-art methods (36% in [1], 25% in [10]). For text inference, *ViViSnoop* detects passwords (random inputs) with around 55% single-character accuracy, and recover words with 71.4% accuracy using top-1 choices and 100% accuracy using top-10 choices. To infer real-world sentences, *ViViSnoop* achieves top-1, top-5 and top-10 accuracy at 59%, 68% and 74% so that there is almost no difficulty in understanding the main user input. We also validate that *ViViSnoop* still performs effectively under various settings.

II. Video Processing Preliminaries

In this section, we introduce two well-established computer vision techniques which are used to extract vibrations from captured videos by *ViViSnoop*.

Phase-based optical flow estimation. This technique extracts and characterizes object motions in a video stream using phase information. For any area of interest in one frame, the phase variation is learned from the changes between the consecutive frames. While no movement occurs during the elapsed time between consecutive frames, the area remains the same and the phase is constant. Otherwise, the phase variation reflects the motion captured by the video.

Steering pyramid decomposition. This is a standard technique to perform texture analysis and synthesis. It is used to compute subtle vibration incurred in this work by extracting phase variations. Because multiple objects in one image may have different sizes and cannot be analyzed through a single scale, the steering pyramid decomposition thus decomposes the images according to different scales and orientations. They use certain complex-valued filter banks [11], [12]. Each video frame is thus decomposed into complex-valued sub-bands corresponding to different spatial scales r and orientations θ. Each sub-band at spatial location (x,y) at time t can be represented as a complex image, $A(r,\theta,x,y,t)\cdot e^{\phi(r,\theta,x,y,t)}$, where $A(r,\theta,x,y,t)$ is its local amplitude and $\phi(r,\theta,x,y,t)$ is its local phase at this sub-band. That is, steering pyramid decomposition helps to separate the phase from the amplitude of each local sub-band along the corresponding scale r and orientations θ. Therefore, the current frame i and the reference frame i_0, the phase variation at this sub-band can be extracted as

$$\Delta\phi(r,\theta,x,y,i) = [\phi(r,\theta,x,y,i) - \phi(r,\theta,x,y,i_0)] \bmod 2\pi. \quad (1)$$

This phase variation is approximately proportional to the displacement of the image objects and can be used to detect motions in video frames [13], [14].

III. *ViViSnoop* Overview

In this section, we present the adversary model and the high-level attack framework of *ViViSnoop*.

Adversary model. We consider a victim user with a laptop, desktop, tablet or even a phone. She types her input using a physical keyboard or a virtual keyboard (on the touchscreen of the tablet/phone) placed on a desk. The adversary appears in her proximity (in a range of feet or meters) and is equipped with a commodity camcorder. Such scenarios are very common in public spaces and office spaces. For example, Figure 1 illustrates three typical scenarios: (1) the victim uses his/her laptop (tablet/phone) at a cafe, hotel lobby, public library, museum or other public space where someone with a smartphone is nearby; This is a commonplace scenario in our daily life. (2) the victim uses one of many desktops in an office, lab, campus, or other public space, while another desktop is used by the adversary; (3) the victim uses her laptop (tablet/phone) in public spaces where a surveillance camera is deployed and the adversary has access to the recorded videos. We assume that the desk is used exclusively by one victim user. The extension to multiple persons is left for future studies.

The attacker cannot go too close (within decimeters) to the typing victim because she is alert to shoulder-surfing attacks; The attack camcorder is built-in, such as a phone rear-camera, a front PC webcam and a surveillance camera, and is thus hardly noticeable when it is in use at a distance. The attacker can record the surroundings of the victim user, but has no need (ability) to record the victim's typing, fingertips, or hand movements. The captured video contains the desk surface with certain texture patterns. The attack requires nothing from the victim's side: there is no need to install malware or compromise the victim's device. Note that under these assumptions, all other inference attacks are not applicable except VISIBLE [1]. The difference between ours and VISIBLE is that *ViViSnoop* works for any physical and virtual keyboard, whereas VISIBLE targets the virtual keyboard of a tablet. Moreover, VISIBLE requires the victim user to hold the tablet with a holder while the backside is recorded whereas in *ViViSnoop*, the keyboard is placed on the desk and the desk surface is recorded.

Attack framework. *ViViSnoop* spies on what the victim user is typing without visually seeing it. It infers her typed inputs from the video capturing the desk on which the keyboard is placed. Figure 2 depicts its high-level ideas in the four steps.

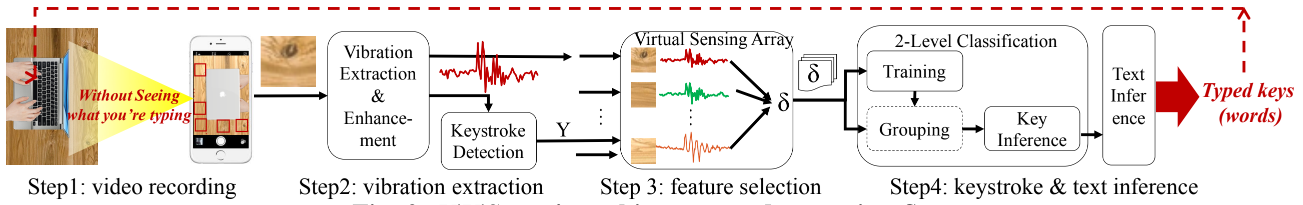

Without Seeing what you're typing
Vibration Extraction & Enhance-ment
Keystroke Detection
Y
Virtual Sensing Array
δ
2-Level Classification
Training
Grouping
Key Inference
Text Inference
Typed keys (words)
Step1: video recording
Step2: vibration extraction
Step 3: feature selection
Step4: keystroke & text inference

Fig. 2: *ViViSnoop*'s architecture and operation flow.

Step 1: Video recording. The adversary first records a video capturing the desk where the victim user is typing on the keyboard. The video does not capture user movements or finger positions on the keyboard.

Step 2: Vibration extraction. The adversary then extracts vibration signals from the captured video frames. We apply the existing video processing techniques for motion detection and perform additional mechanisms (such as noise filtering and vertical projection) to enhance the vibration quality. We also perform keystroke detection to determine whether typing occurs. The rest steps run only when a keystroke is detected.

Step 3: Feature selection. For the third step, the adversary selects features from the extracted vibration signals. However, we find that those vibrations caused by different keystrokes (positions) are still too close to distinguish one from the other. The existing approaches fail to locate features for reliable keystroke distinction. To this end, we devise a novel virtual array sensing technique that extracts multiple vibrations at different positions of the desks caused by the same keystroke. By deriving the difference (Δ) among those vibrations for one keystroke, we can create location-sensitive signatures as the feature vector.

Step 4: Keystroke and text inference. Finally, we infer the typed keys and texts using a two-level classifier. Instead of individual keystroke detection, we first do a coarse-grained classifier for several key groups each of which contains keys difficult to distinguish. Afterwards, we infer the specific key out of its group by leveraging meaningful combinations of characters using a dictionary and the relationship between adjacent words in a sentence. This mechanism has proven to be efficient and robust with a small set of training samples and low-accuracy individual key detection. The first-level classifier is trained using the dataset from the attackers, who imitate the victim's input or from the victim, *e.g.*, her typing input of several sentences is captured when the attacker wanders around her for a short time.

IV. Technical Details

We next elaborate on technical details at each step. In this section, we consider one scenario setting as shown in Figure 3a, where an Apple Bluetooth wireless keyboard is placed on a wooden table (1.25m wide) in a meeting room and the adversary uses the rear camera of an iPhone 6S which is mounted on a tripod around 1.5 meters away. More scenarios are considered in the evaluation (§V).

A. Video Recording and Vibration Extraction

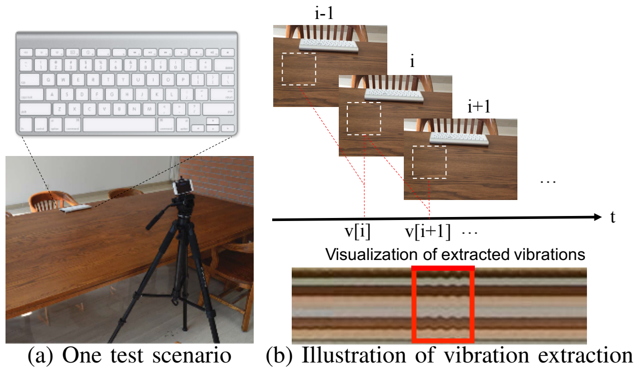
Our attack works under the principle that each keystroke causes certain vibration on the desk though they are subtle and visually unnoticeable to the naked eye. This phenomenon has been validated and exploited for keyboard inference attacks [10], but the vibrations are collected and decoded by the accelerometer at the victim's smartphone nearby (about 2 cm apart). Contrary to the prior work, *ViViSnoop* extracts vibrations from videos captured at a distance (several meters away). Thus the first challenging task is to determine whether and how we can extract suitably strong vibration signals through video processing.

To extract subtle motions in the video, we apply the classic computer vision processing techniques introduced in §II. In particular, we first select one sub-area in the video frame which corresponds to the desk surface containing certain texture patterns. We then apply Steering Pyramid Decomposing technique to retrieve the phase variances between sequential frames and the reference frame along different scales and orientations, as illustrated in Figure 3b. For each subband at spatial location (x,y), we calculate its phase variance as Eqn.(1). We then sum up all the pixel-level phase variances to estimate the total phase variance of the sub-area which approximates the motion contained. The texture is required to extract the pixel-level motion information because no changes can be extracted when all the pixels are the same regardless of movement or not. This requirement is easy to meet since most tables (such as wooden desks) contain natural textures. The texture factor is discussed later in §VII. Eventually, we obtain the raw vibration as a weighted sum of phase variance at different scales r and orientations θ as

$$V[i] = \sum_{r,\theta} \phi(r,\theta,i) = \sum_{r,\theta}\sum_{x,y} A(r,\theta,x,y,t)^2 \Delta\phi(r,\theta,x,y,i). \quad (2)$$

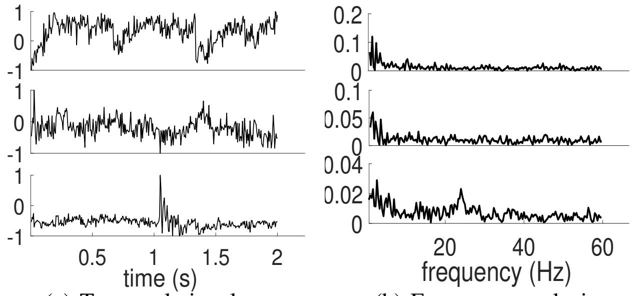
where the local amplitude of $A(r,\theta,x,y,t)$ represents the measure of texture strength. This process is similar to motion extraction in [1], [13]. The bottom plot of Figure 3b visualizes the vibration extracted using the visualization approach (magnified by 15 times) [15]. Figure 4a (top) plots the observed vibrations when the key '`q`' is pressed. Clearly, although we are able to extract vibrations, they seem too subtle and noisy.

To enhance the extraction quality, we perform vertical projection and noise filtering based on the vibration characteristics. Contrary to the prior work where the vibration is directly collected by the accelerometer at a very close smartphone [10], the vibration signal is *implicitly* extracted from the videos captured at a distance and thus suffers from more sources of noise in practice. Its quality is affected by many factors including illumination conditions, recording distance and angles, camcorder capability and settings such as frame rate,



(a) One test scenario (b) Illustration of vibration extraction

Fig. 3: Vibration extraction from the video captured at a distance.



(a) Temporal signals (b) Frequency analysis

Fig. 4: Vibration extracted for one keystroke 'q'. Three rows represent the original vibration signal computed without decomposition (top), the horizontal decomposition (middle) and the vertical decomposition (bottom).

resolution, ISO and zoom configurations. Moreover, *ViViSnoop* targets a realistic setting where the keyboard is firmly held at a certain position as a holder is used for the tablet in VISIBLE [1]. Consequently, the tiny disturbance caused by environment factors such as human walking, talking and light variation further degrades the extracted vibration quality.

Vertical projection. We find that the vertical decomposition is closely relevant to the vibration caused by keystrokes. Figure 4a plots the decompositions along the horizontal and vertical orientations. Apparently, no clear vibration signals are observed in the horizontal one. This is not hard to understand from the viewpoint of the mechanical process. In practice, each key press is almost vertically downward; Consequently, it incurs the strongest disturbance in the absolute upward direction. However, as the camera may be placed at arbitrary recording angles, the question is how to derive the upward vibration from video processing.

We reveal that a vertical projection (decomposition) extracted from the 2D video frames is proportional to the absolute upward vibration. Let us assume the camera is placed at any recording angle η relative to the upward direction. We retrieve the vertical decomposition from video processing as $V_{vertical}$. As the camera has an angle view at η, the projection from $V_{vertical}$ to V_{upward}, the one along the absolute upward direction is $V_{vertical} \cdot \cos\eta$. When the camera stays still, η is almost constant. Consequently, it is reasonable to use the vertical composition extracted from video processing to represent the absolute upward vibration. That is, the vibration signal extracted is the local motion signals decomposed in the vertical orientation at all scales,

$$V_{vertical}[i] = \sum_r \phi(r, \pi/2, i). \quad (3)$$

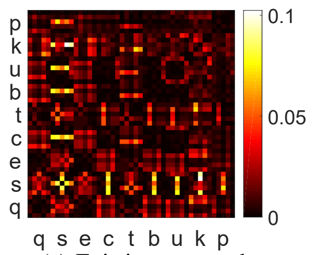
Noise Filtering. We further find that the vibration signals are polluted by certain low frequency noises not caused by keystrokes. Figure 4b shows the FFT coefficients in the frequency domain. We can see that the frequency corresponding to the vertical vibration caused by keystrokes is about 23Hz. It is determined by its inherent mechanical characteristics. In contrast, the low frequency ones are background noise caused mainly by the environment such as floor vibration due to human walking, flicker noise in video capturing [16] and slight light variation. To this end, we run a high pass filter to remove low frequency noises and strengthen the vibration signal quality.

Keystroke detection. We then perform keystroke detection to determine whether a keystroke occurs. A keystroke is detected as long as the certain signal is significantly larger than the background noise. We first calculate the energy threshold of background noise signals (its mean plus $\alpha\times$ standard deviation as an upper bound estimate). We use the sliding window and calculate whether the signal energy is larger than $\beta\times$ noise energy threshold. In this work, we empirically set $\alpha = 4$, $\beta = 3$ and all keystrokes are reliably (100%) detectable in the evaluation.

B. Feature Selection

Given the vibration signals extracted from a contact-free visual channel, the next challenging task is to find the proper features which enable fine-grained keystroke distinction. Ideally, each different keystroke should generate unique signatures for the subsequent keystroke and text inference. Unfortunately, despite the aforementioned enhancement efforts in §IV-A, we find that the vibrations caused by different keystrokes are too close to distinguish one from the other.

We apply the common approaches of feature selection and show all work poorly. In particular, we derive the vibration signal in the time-domain, frequency-domain and spectrum-domain and retrieve its temporal, spectral and cepstral features such as kurtosis (peak) values, root-mean-square values, FFT coefficients, mel-frequency cepstral coefficients (MFCCs) as the feature vector. These features have been widely used in previous keystroke inference studies, *e.g.*, [10], [17], [18]. We measure the distance of all the (anchoring, testing) pairs where each key uses 5 anchoring samples and 5 testing samples. We calculate the correlation coefficients of feature vectors to measure the similarity of two sequences. Due to space limitation, we present only the results for 9 keys and the results for all the alphabetical inputs from 'a' to 'z' to convey similar findings. Figure 5a shows how close each keystroke is to another. A nice keystroke signature feature should exhibit small self-distance but large distance from others. However,


(a) Existing approaches

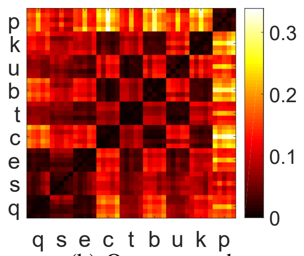
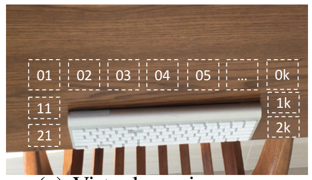
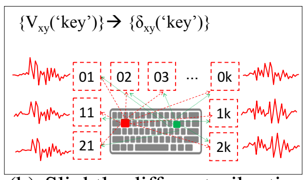

(b) Our approach

Fig. 5: Comparison of feature selection.


(a) Virtual sensing array



$\{V_{xy}(\text{'key'})\} \rightarrow \{\delta_{xy}(\text{'key'})\}$

(b) Slightly different vibrations observed at various positions

Fig. 6: Feature extraction through a virtual sensing array.

we can see that most keys are very close to others and it is difficult if not impossible to identify unique keystrokes.

Virtual sensing array. To address this, we devise a novel virtual array sensing technique that extracts multiple vibrations at different positions of the desks caused by the same keystroke. Figure 6 illustrates our idea. We notice that every visual object can be taken as a "*sensor*" to extract the vibration caused by the keystroke. We thus leverage video processing of multiple visual texture objects (sub-areas of one frame) to measure vibrations observed at different positions, without the need to deploy real sensors there. We further harness the tiny distinction between those vibrations to make each keystroke more "*directional*".

Specifically, for each keystroke, we collect a set of vibration signals V_{xy}(`key`), each corresponding to one observation extracted through the visual object at position (x,y). Figure 6b shows several vibration samples observed at different locations. We notice that they are slightly different. This is because when the vibration propagates, the energy will also propagate and dampen with it [19]. Thus it can be exploited to derive the relative positions of the keystroke and the sensor. For example, for the keystroke of `q` (closer to the left side), all the left sensors at $01, 11, 21$ observe different vibrations from those right sensors at $0k, 1k, 2k$. These distinctions help to indicate that the keystroke is closer to the left rather than the right. In this way, by comparing the tiny difference between the vibrations at the two sensing positions, we can roughly locate the relative position of one keystroke. Finally, through multiple pairwise comparison with virtual array sensing, we can derive a position signature for each keystroke.

We thus extract the keystroke feature as a vector of relative energy. Each relative energy is defined as

$$\delta_{i,j}('key') = \frac{e_i}{e_j} = \frac{||V_{xi,yi}('key')||^2}{||V_{xj,yj}('key')||^2}, \quad (4)$$

which represents the difference observed by two visual sensors at position i and j. Note that the number of virtual sensors depends on the selection of image objects in one frame. Ideally, there is no cost to increase the number of sensors as long as each sub-area contains basic texture pattern for vibration extraction. However, in practice, the extraction using a smaller subarea is less accurate as fewer pixels are used. We assess the impact of the sensor number and their layout in the evaluation. We find that our approach is quite effective and reliable as long as the number of sensors is large enough (*e.g.*, using sensors at two rows).

Figure 5b shows a large improvement in the effectiveness of the proposed feature selection. We calculate the distance among the same 9 keys, each using a vector of relative energy. The results for the other keys are similar. Clearly, we can see that this feature vector has greatly enhanced the distinction among different keystrokes. Some keys like `q`, `c` and `q` are obviously distinguishable. We also notice that some keys, like `q`, `s` and `e` are still hard to differentiate from each other while they are distinguishable from other keys. We will address this in the next inference step.

C. Keystroke and Text Inference

In the final step, we use the vibration features at multiple desk positions extracted from one video recording to infer typed keys and texts. Like most prior studies, we follow the standard approach that first uses a classifier to detect a candidate key as the typed key input and then to exploit the relationship of adjacent characters in one word and the one of the words in one sentence to infer the typed texts. To train the classifier, we use the dataset collected by the attacker intimating the victim's input, or directly from the victim user when the attacker has a chance to capture the typing input for a short while (*e.g.*, the attacker wanders around and stays close for a minute or so while the victim user is typing).

However, our study reveals that this approach suffers from one problem. Its inference accuracy counts heavily on individual keystroke detection accuracy which is sensitive to the training dataset size. Most prior studies perform poorly with a small training set (*e.g.*, < 20) and usually require 50-100 or more training samples for each keystroke in their experiments. Nevertheless, it is challenging or even unrealistic to obtain a large number of training samples similar to the victim's typing.

To enhance the inference efficiency and robustness, we propose a two-phase classifier. We first cluster some close keystrokes into groups and run a coarse-grained classification among the groups. We then infer a single keystroke out of its group by leveraging linguistic relationships among characters and words. This design is driven by our observation that most keystroke detection errors are not random. Take Figure 5b as an example, the input of `q` is mistakenly detected as `s` and `e`, but not any arbitrary one. The main reason is the most physically adjacent keystrokes have similar features which are not sufficient to distinguish each other but are often distinct from other keystroke sets. Intuitively, we expect that grouping helps to make a nice knob between individual keystroke detection and the use of linguistic relationships among multiple keystrokes. When the ambiguity raised by grouping is addressed by text inference at the second phase, it can potentially boost efficiency, robustness and accuracy in keystroke inference.

We next present how to cluster keystrokes and adapt the keystroke and text inference to the groups.

Clustering. Clustering aims to achieve a good tradeoff between detection accuracy at the first phase and the additional burden raised at the second recognition phase. The larger the number of groups is, the lower the detection accuracy but the lower recognition burden is, and vice verse. In an extreme case, one group means no keystroke detection (100% accurate) and the inference is mainly through random guess. Another extreme case is one key per group where no clustering is used, and the inference runs as usual. In *ViViSnoop*, we run a recursive algorithm to locate a reasonable number of clusters. We start with one and increase the number of clusters until the group detection accuracy does not degrade too much. To classify them, we select Linear Discriminant Analysis (LDA) classifier because it has been applied widely in small-size dataset [20].

Inference of labelled sequences. We further infer the inputs given a sequence of possible inputs. We extend the original keystroke input to its grouping label. For example, let us assume 'd' belongs to G_3 ("d" $\rightarrow G_3$), 'a' $\rightarrow G_1$, 'r' $\rightarrow G_3$, 'k' $\rightarrow G_5$. Hence, the word "dark" can be expressed as "$G_3G_1G_3G_5$".

Afterwards, we further infer the matched texts and finalize keystroke inputs based on grouping labels. It is more challenging because clustering creates more choices for word matching. For example, the constraint labels for "dark" and "hard" are both $G_3G_1G_3G_5$. To address this, we adopt the long short-term memory (LSTM) language model [21], [22]. LSTM is a kind of recurrent neural network (RNN), which decides the current output based on the current input and historical outputs. Given the labelled sequence seq_t and previous words $w_1, w_2, \ldots, w_{t-1}$, we derive the probability of next word w_t at time t as

$$P(w_t|\text{seq}_t, w_1, \ldots, w_{t-1}) = \begin{cases} 0, & \text{tag}(w_t) \neq \text{seq}_t \\ P(w|h_{t-1}, \Theta), & otherwise \end{cases} \quad (5)$$

where Θ is a parameter of LSTM language model and h_{t-1} is a vector representing the subsequence $w_1, w_2, \ldots, w_{t-1}$. We further run the beam-search algorithm to locate a sequence of decisions. At every step, a fix-sized beam is maintained for potential candidates in the descending order of sequence probabilities. Once the beam is full, the one with the highest probability will survive. At the next step, an incoming word will be integrated every word survived in the previous beam. Beam-search is proven to achieve reasonably good top-k results [23].

V. Evaluation

We evaluate the performance of *ViViSnoop* through extensive experiments in three aspects: (1) effectiveness of design components in micro-benchmark experiments, (2) accuracy of single keystroke inference and the impact of various environmental factors and *ViViSnoop*'s configurations, and (3) overall accuracy of text inference.

Experiment setting. The attack side uses an iPhone 6s which has a 12-megapixel rear camera supporting 1080p HD video recording up to 60fps and slo-mo video recording up to 240fps. The victim side uses three physical keyboards (by Dell, Arteck and Apple) and one virtual keyboard displayed on an iPad Mini 1 (13.4 cm $\times$ 20 cm). We consider only 26 alphabetical keys in this evaluation. By default, we use the Apple Bluetooth keyboard placed at the center of the desk edge and the iPhone 6S placed 1m away, at a roughly 5^o degree which can capture the desk surface surrounding the keyboard. The basic experiment setting is shown in Figure 3a. We also evaluate the impacts of environmental factors such as different keyboard placement, recording distances and angles and illumination conditions, as well as the *ViViSnoop*'s parameters including the recording rate, number of virtual sensors, cluster number and the size of training samples. By default, the camcorder runs at 60fps and we use two-layer virtual sensors. The training uses 10 strokes for each key, 260 keystrokes in total from the victim user. We also test with more participants and use their samples to train the classifier for others.

Keystroke Detection	**Single Keystroke**	**Clustering** (5 groups)	**Word**	
			Top-1	Top-10
100%	50%	89%	71.4%	100%

TABLE I: Micro benchmark results at each step.

Effectiveness of design components. Table I shows the performance of each step while we test with a 27-word dataset used in prior works [1], [24]. We have four observations. First, *ViViSnoop* is able to detect the occurrence of keystrokes (Step 2) without any error; this is because *ViViSnoop* can extract good quality vibrations through video processing. Second, *ViViSnoop* achieves an average accuracy of 50% in single keystroke inference. Moreover, it outperforms prior studies ([1]: 36%, [10]: 25.9%). Figure 7 compares their single-key inference accuracy. It indicates that feature selection through the virtual sensing array is quite effective and offers more fine-grained keystroke distinction. Third, we further improve the accuracy to 89% as we cluster certain error-prone and close keystrokes into clusters (here, we use 5 clusters). Later we also examine how clustering affects the first-phase accuracy (Figure 8c). Finally, we infer the typed words with the help of the LSTM model. It achieves 71.4% and 100% accuracy using the top-1 and top-10 choices, respectively.

A. Single Keystroke Inference Accuracy

We further examine single keystroke inference accuracy through an extensive evaluation of its impact factors. Note that no studies achieve extremely high accuracy at this stage (somehow resolved by the use of the language model later), 50% is considered to be a very good benchmark.

• **Design factors.** We first investigate three design choices of the visual sensor numbers, the recording frame rate, as well as the size of the training set. In theory, there are numerous

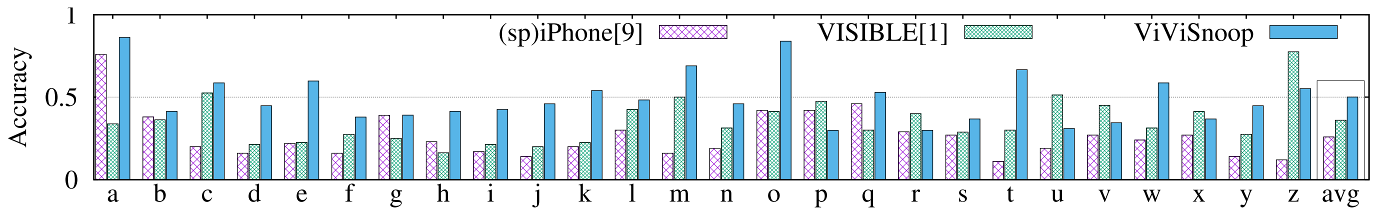
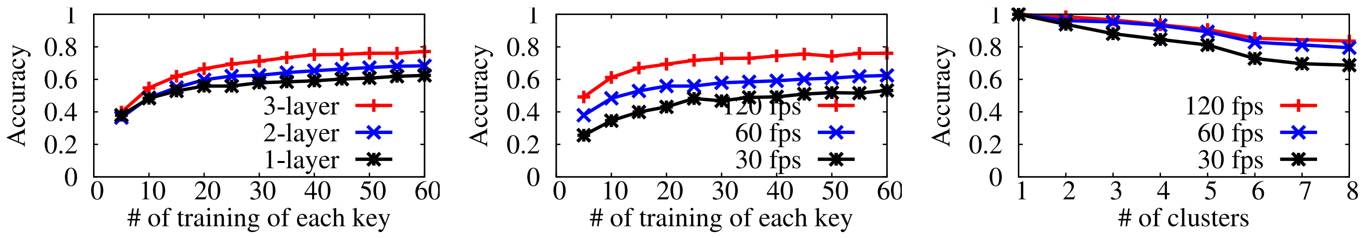
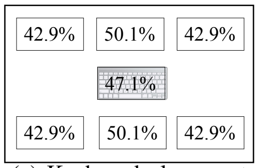


Fig. 7: Comparison of single-keystroke inference accuracy.



(a) Single: number and layout of sensors (b) Single: frame recording rate (c) Group: frame recording rate

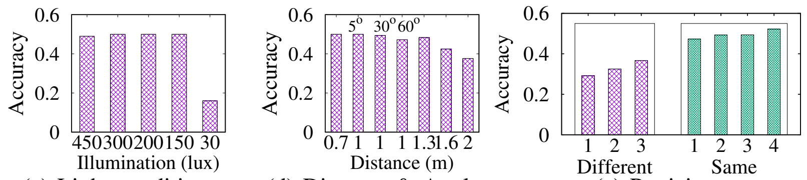
Fig. 8: Single keystroke inference accuracy and group accuracy under various design factors.



(a) Keyboard placement

Keyboard	Acc(%)
Dell	46.3%
Arteck	50.5%
Apple	50.1%
iPad	51.8%

(b) Keyboards



(c) Light conditions (d) Distance & Angle (e) Participants

Fig. 9: Single keystroke inference accuracy under various environmental factors.

visual sensors on the desk. We align these "sensors" row by row and each row is referred to as a "layer". Each layer usually contains 8 to 12 visual sensors. We vary the number of sensors from one layer to three layers. The size of training set varies from 5 to 60 for each key, where the total samples for each key is 100. The training set is randomly selected and the rest are used as the testing samples. We also test with three common recording rates (30fps, 60fps and 120fps), which are supported by most smartphone and monitoring cameras. We expect to examine the impact of video quality.

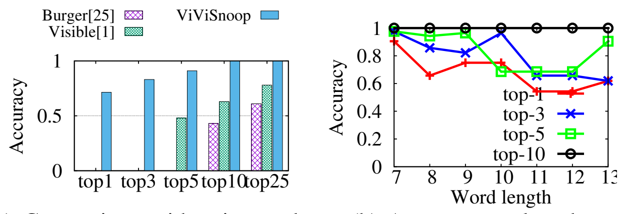
Figure 8a and 8b show their impacts on single keystroke inference accuracy. Clearly, with more visual sensors, a larger training set and a higher frame rate can improve accuracy. In particular, we make three observations. First, given a larger training set, the inference accuracy is greatly improved. With 60 training samples, *ViViSnoop* achieves 60–78% accuracy (60fps) and 50% accuracy (30fps). However, a large training set may make our attack impractical. With fewer training samples, the accuracy decreases but is still quite effective (compared with the existing studies). Given 5 training samples, the accuracy remains 35–40% (60fps, 1-3 sensor layers) and 25–50% (30-120fps, 2 sensor layers). In the following experiments, we use 10 training samples by default. Second, more sensors help to boost inference accuracy, although the help is less obvious with a larger training set. On the other hand, it implies that our virtual sensing array can compensate for insufficient training samples by providing finer-grained keystroke features. We choose 2-layer sensors afterwards. Last, the video recording rate also plays a critical role. This is because information is lost when we record at a low sampling rate. We also evaluate the impact on group accuracy in Figure 8c. We can see that the accuracy gap caused by different recording rates can be compensated by the clustering technique. This helps *ViViSnoop* to be more resilient to less accurate single keystroke inference. In this study, we choose 60fps as the default frame rate which can be met by most commodity camcorders and select 5 as the final cluster number with 89% group inference accuracy.

We next examine the impacts of environmental factors such as keyboard placement, keyboard types, light conditions, recording distance and angles, and the number of participants involved.

• **Keyboard placement.** We place the keyboard at different positions of desktop including the center edge, the center and the side of the desk. Figure 9a shows that *ViViSnoop* works for any position though the accuracy is slightly higher at the center of the desk edge.

• **Keyboard types.** Table 9b shows the average accuracy of three physical keyboards of Dell wired keyboard (KB216), Arteck stainless steel portable bluetooth keyboard and Apple Bluetooth keyboard, and a virtual keyboard on an iPad Mini 1. The accuracy varies from 46% to 51%, which indicates that *ViViSnoop* is widely applicable. We find that the accuracy of the Dell keyboard is a bit lower than the other three because the alphabet keys are all mainly on the left side, and its layout makes it a little harder to recognize.

• **Illumination conditions.** We find that *ViViSnoop* works quite reliably under normal light conditions at lab (450 *lux*),



(a) Comparison with prior studies (b) Accuracy vs. length

Fig. 10: Word inference accuracy.

Typed: our friends at the university of texas are planning
Recovered: <u>*two*</u> friends at the university of texas are planning
<u>our</u> (top-2)
T: conference on energy economics and finance in february
R: conference in <u>*bloody*</u> economics and finance in <u>*decrepit*</u>
energy(top-5,cloudy→clergy →mzoudi)
T: of next year
R: of next year

Fig. 11: One example of email sentence recovery.

office (300 *lux*), and Starbucks (200/150 *lux*). Here, the accuracy declines to 16% only under dim illuminations (30 *lux*). This matches with our expectation because the video fails to capture enough details without sufficient illumination.

• **Recording distances and angles.** We test five distances at 0.7m, 1m, 1.3m, 1.6m and 2m and use zoom-in when the distance is larger than 1.3m. We also test with three angles $5^o, 30^o$ and 60^o at 1m and the accuracy fluctuates slightly at 50%, 48.9% and 44.5%. It shows that *ViViSnoop* still works as long as it can capture the desk surface though the accuracy decreases a bit when the captured desk surface is slightly apart from the keyboard at other angles. The distance plays a more critical role. The accuracy degrades from 50% to 37% at a distance of 2m. This reflects the impact of camcorder limits on *ViViSnoop*. When the distance increases, the video quality degrades even with the zoom-in use.

• **Participants.** We also examine the impact of different participants and training sources. We have four participants and each has 10 strokes for each key as training samples and 30 strokes for test. Figure 9e shows the results using the mimic training from different users and the training from the same victim users. In a different category, we use the training set from 1, 2, 3 users to predict the other 3, 2, 1 users' typed inputs Clearly, *ViViSnoop* works consistently for different victims and the accuracy in the same category remains around 49%. The accuracy degrades using the training samples from others but it can be compensated as the number of attackers increases. When 3 participants are involved to recover one victim's typing, the accuracy increases to 37%.

B. Overall Text Inference Accuracy

We use various text inputs including passwords (random inputs), words and sentences to evaluate the overall accuracy.

• **Password.** Password inference is one of the most threatening and challenging attacks, because passwords have no regular patterns and in most cases are totally random. We test with the password dataset published by SkullSecurity [25]. Table II shows, for 50K passwords, the average accuracy of each character is 55.8%, comparable to (slightly better than) the individual keystroke inference (§V-A). This implies that LSTM is of little help in improving the inference accuracy for random inputs.

• **Word.** We use 27 words with length varying from 7 to 13 to evaluate the word inference accuracy as Berger [24] and VISIBLE [1] also do. Figure 10 shows that our approach outperforms prior studies. Our first choice (top-1) accuracy achieves 71.4% in average, which is even higher than their top-5 accuracy (still below 50% in [24] and [1]). Moreover, we can retrieve all the words within top-10 choices. We further show the accuracy with regard to the word length in Figure 10b. Our approach has lower accuracy for longer words because more grouping labels are introduced to longer words and incur more sources of errors.

#. passwords	20	100	1000	10000	20000	50000
Avg-accuracy (%)	55.6%	61.0%	57.9%	56.6%	56.3%	55.8%

TABLE II: Inference accuracy of password characters.

• **Real-world sentences.** We finally use the Enron Email Dataset (600K emails) [26] to evaluate *ViViSnoop*'s impact in real-world. We select the same sentences used by VISIBLE [1] and Figure 11 shows one recovered result. The first recovered row shows the top-1 choice word and the red one is a wrong one; The one in the second row is the top-n choice. In this example, our top-1 choice recover 17 words out of 20 words, with an accuracy of 85%, which is much higher than VISIBLE [1], 10%. For top-5 choice accuracy, ours can recover 19 words (except "february") with an accuracy of 95%. We further test with more sentences. We randomly select 100 sentences (1845 words) out of the dataset. *ViViSnoop* achieves the average accuracy of 59.3%, 68.4%, 74.9% using top-1, top-5, top-10 choices on average (Table III). This indicates that *ViViSnoop* imposes a serious threat to information leakage and user privacy since even ordinary people have no difficulty in understanding the main idea of the emails.

Top-1	**Top-3**	**Top-5**	**Top-10**
59.3%	63.6%	68.4%	74.9%

TABLE III: Inference accuracy of words in sentences.

VI. Related Work

We now introduce the prior work most related to *ViViSnoop*.

Video-assisted attacks. *ViViSnoop* belongs to this category which applies computer vision techniques to infer user inputs from the video captured. However, most prior studies require to visually capture the user input results or their typing processing. They retrieve the content on the screen through the surrounding reflections like user eyes [2], tea pots [3], and sunglasses [5], or infer the user inputs through hand movement [7] or fingertips [6], [8], [9] during the typing.

The most relevant work is VISIBLE [1] which captures the backside of the tablet and does not require capturing the victim user's input process. However, our work differs in three aspects. First, we target any (physical or virtual) keyboard placed on the desk. Second, we extract indirect vibrations on the desk rather than direct motions caused by keystrokes. This is more challenging to ensure good quality. Third, we develop a suite of video processing, feature selection and classification techniques tailored to the *ViViSnoop* scenarios.

Non-video based attacks. Another most relevant work is (sp)iphone [10] which also exploits vibrations for keyboard inference attack. But it uses the accelerometer from the victim phone closely near the keyboard. It is less challenging to extract vibrations; Moreover, this requires to install a malware on the victim's phone and allow unprotected network access to disseminate the sensing data to the attacker. There are other inference attacks that exploit non-video source like the accelerometer and/or gyroscope data (*e.g.*, [27], [28]) and acoustic signals (*e.g.*, [17], [24], [29]). But their attack models are the same as [10], and are not applicable in our scenarios.

VII. Discussion and Conclusion

In this paper, we propose *ViViSnoop*, a video-assisted keystroke inference attack which reveals the user's typing on a keyboard placed on the desk through recording a seemingly irrelevant video, without visually seeing the typing process. Instead, it infers the keystroke through the vibrations on the desk retrieved from the captured video. It is much less noticeable as video recording can be stealthily done with no stringent constraints on angles and distance. It is threatening and ready to launch as it does not require any malware or compromise on the victim side. We thoroughly evaluate its performance and show high accuracy and probability to infer passwords, words and sentences.

There are several limitations and countermeasures against *ViViSnoop*. First, it performs poorly at a large distance. This attack can be avoided when the user is alert to anyone or any possible camcorder nearby. Second, it relies on incurred vibrations on the desk and certain textures to extract them. The victim user can easily avoid such attacks without placing the keyboard on the desk. It would be safe to use other desks without incurring any vibrations, or use a desk without any visual patterns. Third, we consider only 26 alphabet letters in this work. Other keys like numbers, space, and combinations with special keys (*e.g.*, shift, ctrl) will raise a more challenging task which is our ongoing work.

References

[1] J. Sun, X. Jin, Y. Chen, J. Zhang, R. Zhang, and Y. Zhang. Visible: Video-assisted keystroke inference from tablet backside motion. In *NDSS*, 2016.

[2] M. Backes, M. Dürmuth, and D. Unruh. Compromising reflections-or-how to read lcd monitors around the corner. In *IEEE S&P*, 2008.

[3] M. Backes, T. Chen, M. Duermuth, H. Lensch, and M. Welk. Tempest in a teapot: Compromising reflections revisited. In *IEEE S&P*, 2009.

[4] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from video. In *IEEE S&P*, 2008.

[5] R. Raguram, A. M. White, D. Goswami, F. Monrose, and J.-M. Frahm. ispy: automatic reconstruction of typed input from compromising reflections. In *ACM CCS*, 2011.

[6] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm. Seeing double: Reconstructing obscured typed input from repeated compromising reflections. In *ACM CCS*, 2013.

[7] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha. Beware, your hands reveal your secrets! In *ACM CCS*, 2014.

[8] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao. Blind recognition of touched keys on mobile devices. In *ACM CCS*, 2014.

[9] Q. Yue, Z. Ling, W. Yu, B. Liu, and X. Fu. Blind recognition of text input on mobile devices via natural language processing. In *Proceedings of the 2015 Workshop on Privacy-Aware Mobile Computing*, pages 19–24. ACM, 2015.

[10] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *ACM CCS*, 2011.

[11] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable multiscale transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, 1992.

[12] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

[13] A. Davis, M. Rubinstein, N. Wadhwa, G. Mysore, F. Durand, and W. T. Freeman. The visual microphone: Passive recovery of sound from video. *ACM Transactions on Graphics*, 33(4):79:1–79:10, 2014.

[14] T. Gautama and M. M. Van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, 13(5):1127–1136, 2002.

[15] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman. Phase-based video motion processing. *ACM Trans. Graph.*, 32(4), 2013.

[16] E. Milotti. 1/f noise: a pedagogical review. *arXiv preprint physics/0204033*, 2002.

[17] T. Zhu, Q. Ma, S. Zhang, and Y. Liu. Context-free attacks using keyboard acoustic emanations. In *ACM CCS*, 2014.

[18] J. Wang, K. Zhao, X. Zhang, and C. Peng. Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization. In *ACM MobiSys*, 2014.

[19] A. A. Shabana. *Theory of vibration: an introduction*, volume 1. Springer Science & Business Media, 1995.

[20] A. Sharma and K. K. Paliwal. Linear discriminant analysis for the small sample size problem: an overview. *International Journal of Machine Learning and Cybernetics*, 6(3):443–454, 2015.

[21] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[23] Y. Zhang and S. Clark. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151, 2011.

[24] Y. Berger, A. Wool, and A. Yeredor. Dictionary attacks using keyboard acoustic emanations. In *ACM CCS*, 2006.

[25] Skullsecurity. https://wiki.skullsecurity.org/.

[26] Enron email dataset, 2015. https://www.cs.cmu.edu/ẽnron/.

[27] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang. Accessory: password inference using accelerometers on smartphones. In *HotMobile*, 2012.

[28] Z. Xu, K. Bai, and S. Zhu. Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors. In *WiSec*, 2012.

[29] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE S&P*, 2004.