Skype & Type: Keyboard Eavesdropping in Voice-over-IP

STEFANO CECCONELLO, University of Padua, Italy
ALBERTO COMPAGNO, Cisco Systems, France
MAURO CONTI[*], University of Padua, Italy
DANIELE LAIN[†], ETH Zurich, Switzerland
GENE TSUDIK, University of California, Irvine, USA

Voice-over-IP (VoIP) software are among the most widely spread and pervasive software, counting millions of monthly users. However, we argue that people ignore the drawbacks of transmitting information along with their voice, such as keystroke sounds – as such sound can reveal what someone is typing on a keyboard.

In this paper we present and assess a new keyboard acoustic eavesdropping attack that involves VoIP, called *Skype & Type* (*S&T*). Unlike previous attacks, *S&T* assumes a weak adversary model that is very practical in many real-world settings. Indeed, *S&T* is very feasible, as it does not require: (i) the attacker to be physically close to the victim (either in person, or with a recording device); (ii) precise profiling of the victim's typing style and keyboard; moreover it can work with a very small amount of leaked keystrokes. We observe that leakage of keystrokes during a VoIP call is likely, as people often "multi-task" during such calls. As expected, VoIP software acquires and faithfully transmits all sounds, including emanations of pressed keystrokes, which can include passwords and other sensitive information. We show that one very popular VoIP software (Skype) conveys enough audio information to reconstruct the victim's input – keystrokes typed on the remote keyboard. Our results demonstrate that, given some knowledge on the victim's typing style and keyboard model, the attacker attains top-5 accuracy of 91.7% in guessing a random key pressed by the victim. This work extends previous results on *S&T*, demonstrating that our attack is effective with many different recording devices (such as laptop microphones, headset microphones, and smartphones located in proximity of the target keyboard), diverse typing styles and speed, and is particularly threatening when the victim is typing in a known language.

CCS Concepts: • **Security and privacy → Authentication**; **Side-channel analysis and countermeasures**.

1 INTRODUCTION

Circumvention of cryptographic-based data protection techniques usually requires compromising one of the end-hosts, to be able to capture plain-text before it is encrypted. A more convenient way of capturing plain-text before encryption, thus avoiding the needs of compromising a system, is eavesdropping on physical emanations: observing unintentional leakage of emanations which happen during the regular devices' operations. First discovered in 1943 [20], this mechanism has been proven to be applicable to several types of emanations: electromagnetic [39], visual [10], tactile [36], and acoustic [8, 21].

Convenient targets for physical eavesdropping attacks are I/O peripherals (e.g., keyboards, mice, touch-screens and printers), as their emanations directly leak information on the unencrypted input or output text. In particular, exploitation of keyboard acoustic emanations proved effective in reconstructing the typed input. In this specific class of eavesdropping attacks, an adversary learn what a victim is typing by analyzing the sound produced by the keystrokes. Typically, sounds

[*]Also with University of Washington, USA.
[†]This work was partially done when at University of Padua, Italy.

Authors' addresses: Stefano Cecconello, Department of Mathematics, University of Padua, Italy, stefano.cecconello@math.unipd.it; Alberto Compagno, Cisco Systems, France, acompagn@cisco.com; Mauro Conti, Department of Mathematics, University of Padua, Italy, conti@math.unipd.it; Daniele Lain, Department of Computer Science, ETH Zurich, Switzerland, daniele.lain@inf.ethz.ch; Gene Tsudik, Department of Computer Science, University of California, Irvine, USA, gene.tsudik@uci.edu.

are recorded either directly, using microphones [8, 12, 23, 24, 29, 33, 40, 44, 45], or by exploiting various sensors (e.g., accelerometers [32, 42]) to re-construct the same acoustic information. Once collected, the audio stream is typically analyzed using techniques, such as supervised [8, 23, 24, 33] and unsupervised [12, 45] machine learning, or triangulation [29, 40, 44]. The final result is a full or partial reconstruction of the victim's input.

In the past years, all proposed keyboard acoustic eavesdropping attacks required a compromised (i.e., controlled by the adversary) microphone near the victim's keyboard [8, 12, 23, 24, 29, 33, 40, 44]. However, requiring physical access strongly limits applicability of such attacks, thus reducing their real-world feasibility. For this reason, recent proposals [7, 17] relaxed the physical proximity requirement, exploiting Voice-over-IP (VoIP) applications to move the adversary in a remote-setting scenario. Premise to this attack, called *Skype & Type attack* (or *S&T attack* for short), is the observation that people involved in VoIP calls often engage in secondary activities, such as: writing email, contributing their "wisdom" to social networks, reading news, watching videos, and even writing research papers. Many of these activities involve using the keyboard (e.g., entering a password). VoIP software automatically acquires all acoustic emanations, including those of the keyboard, and transmits them to all other parties involved in the call. If one of these parties is malicious, it can determine what the user typed based on keystroke sounds. Such an adversary is realistic: it is not always the case that two parties engaged in a VoIP call have mutual trust. For example, such a situation might happen between lawyers on opposite sides of a legal case, or negotiators for different parties. Moreover, the pervasiveness of VoIP software provides to an attacker a huge attack surface, hard to achieve with previous approaches. Considering Microsoft Skype alone, the number of active monthly user is about 300 million [5].

Contributions. In this paper, which is an extended version of our work in [17] we make the following contributions:

- We demonstrate *S&T attack* based on remote keyboard acoustic eavesdropping over VoIP software, with the goal of recovering text typed by the user during a VoIP call with the attacker. *S&T attack* can also recover random text, such as randomly generated passwords or PINs. We take advantage of spectral features of keystroke sounds and analyze them using supervised machine learning algorithms.
- We evaluate *S&T attack* over a very popular VoIP software: Skype. We designed a set of attack scenarios that we consider to be more realistic than those used in prior results on keyboard acoustic eavesdropping. We show that *S&T attack* is highly accurate with minimal profiling of the victim's typing style and keyboard. It remains quite accurate even if neither profiling is available to the adversary. Our results show that *S&T attack* is very feasible, and applicable to real-world settings under realistic assumptions. *S&T* allows to greatly speed up brute-force cracking of random passwords. If the target text is not random, *S&T* exploits contextual information (such as the typing language) to recognize words with very high accuracy. Moreover, experiments with Google Hangouts indicate that it is likely susceptible to *S&T attack* as well.
- We show, via extensive experiments, that *S&T attack* works well with different common and inexpensive recording devices, on a great variety of typing styles and speed, and is robust to VoIP-related issues, such as limited available bandwidth that degrades call quality, as well as human speech over keystroke sounds.
- Based on the insights from the design and evaluation phases of this work, we propose a countermeasure to *S&T* and similar attacks that exploit spectral properties of keystroke sounds. Our proposed countermeasure is transparent, does not impact severely the quality

of the voice during the call, and is able to disrupt spectral features — making previous data collected by an adversary useless.

Difference with Preliminary Version. The novel contributions of this work, compared to the preliminary version in [17], lie in a greatly extended experimental evaluation, and in improvements to the performance of *S&T* and of our proposed countermeasure. Regarding the experimental evaluation, our preliminary work considered only laptop keyboards and laptop microphones. We now also consider (i) different victim device setups, such as external membrane and mechanical keyboards; and (ii) different recording devices, such as victims that use a smartphone for their VoIP calls, or use a headset microphone, a typical setup in enterprise. Additionally, we shed light on the impact of typing speed, showing that even for the fastest typist *S&T* is still effective, and greatly expand the number of users in our experiments. Regarding the performance of *S&T*, we improve the recognition of non-random text by exploiting contextual information such as the typing language. Finally, we refine our countermeasure and are now able to successfully prevent *S&T attack* in a transparent way, without impacting the quality of the call.

Organization. Section 2 overviews related literature and state-of-the-art on keyboard eavesdropping. Next, Section 3 describes the system model for our attack and various attack scenarios. Section 4, presents *S&T attack*. Section 5 describes how we collected the data that we then use to evaluate *S&T attack* in Section 6. In Section 7 we show some practical applications of *S&T attack*. Finally, Section 8 proposes some potential countermeasures, and Section 9 summarizes the paper and overviews future work.

2 RELATED WORK

Eavesdropping on keyboard input is an active and popular area of research. This section begins by overviewing attacks that rely strictly on acoustic emanations to recover the victim's typed text and then summarizes results that study eavesdropping on other physical emanations, such as the WiFi signal, and surface vibrations. For a complete treatment of keyboard side channel attacks, we refer to the recent survey in [34].

Attacks Using Sound Emanations. Research on keyboard acoustic eavesdropping started with the seminal paper of Asonov and Agrawal [8] who showed that, by training a neural network on a specific keyboard, good performance can be achieved in eavesdropping on the input to the same keyboard, or keyboards of the same model. This work also investigated the reasons for this attack and discovered that the plate beneath the keyboard (where the keys hit the sensors) has a drum-like behavior. This causes the sound produced by different keys to be slightly distinct. Subsequent efforts can be divided based on whether they use statistical properties of the sound spectrum or timing information.

Approaches that use statistical properties of the spectrum typically apply machine learning, both supervised [8, 23, 24, 33] and unsupervised [12, 45] versions.

Supervised learning techniques require many labeled samples and are highly dependent on: (1) the specific keyboard used for training [8], and (2) the typing style [23, 24]. Such techniques use Fast Fourier Transform (FFT) coefficients and neural networks to recover text that can also be random. Overall, supervised learning approaches yield very high accuracy. However, this comes at the price of strong assumptions on how the data is collected: obtaining labeled samples of the acoustic emanations of the victim on his keyboard can be difficult or unrealistic.

Unsupervised learning approaches can cluster together keys from sounds, or generate sets of constraints between different key-presses. It is feasible to cluster key sounds and assign labels to the clusters by using relative letter frequency of the input language [45]. It is also possible to generate sets of constraints from recorded sounds and select words from a dictionary that match these constraints [12]. Unsupervised learning techniques have the advantage that they do not require ground truth. However, they make strong assumptions on user input, such as obtaining many samples, i.e., emanations corresponding to a long text [45], or requiring the targets to be dictionary words [12]. They are less effective when keyboard input is random.

An alternative approach involves analyzing timing information. One convenient way to exploit timing information is using multiple microphones, such as the ones on mobile phones [29, 40, 44], and analyze the Time Difference of Arrival (TDoA) information to triangulate the position of the pressed key. Such techniques differ mostly in whether they require a training phase [40], and rely on one [29] or more [44] mobile phones.

All the previously cited approaches assume physical proximity with the adversary — this requirement can be relaxed by using VoIP software to acquire keyboard sounds, as demonstrated in [7, 17]. The main difference between our work and [7] lies in a much more extensive experimental evaluation on more realistic scenarios, and in a less obtrusive countermeasure, that we validate through more complete testing.

Attacks Using Other Emanations. Another body of work focused on keyboard eavesdropping via non-acoustic side-channels.

Typing on a keyboard causes its electrical components to emit electromagnetic waves, and it is possible to collect such waves, to recover the original keystrokes [39]. Furthermore, typing causes vibrations of the surface under the keyboard. These vibrations can be collected by an accelerometer (e.g., of a smartphone) [32], or by a camera that captures small movements of the surface [25], and analyzed to determine the pressed keys.

Inter-keystroke timing also leaks information about the pressed keys. The idea is that users require slightly different time to type different sequences of characters, allowing keystroke recovery. This was successfully exploited by observing inter-keystroke timing over SSH connections [37], and from videos of users typing on masked password fields [9].

Analyzing movements of the user's hands and fingers on a keyboard represents another way of recovering input. This is possible by video-recording a typing user [10] or by using WiFi signal fluctuation on the user's laptop [6]. In general, changes in channel state information of wireless signal can also be leveraged to recover typed text in a known language without the need for training [19]. On touchscreen keyboards (e.g., smartphones), the typist's eyes follow the fingers' movement — therefore, tracking eye movements can leak information on the typed text [15].

Finally, heat radiation from hands to the device in use can be captured by thermal cameras, and used to infer keystrokes on PIN-entry devices [43], and standard keyboards [26].

3 SYSTEM AND THREAT MODELS

To identify precise attack scenarios, we begin by defining the system model that serves as the base for *S&T*. Section 3.1 describes our assumptions about the victim and the attacker, and then carefully defines the problem of remote keyboard acoustic eavesdropping. Section 3.2 then presents some realistic attack scenarios and discusses them in relation to the state-of-the-art.

3.1 System Model

The system model we consider in this work is depicted in Figure 1. We assume that the victim has a desktop or a laptop computer with a built-in or attached keyboard, i.e., **not** a smartphone or a tablet-like device. Hereafter, it is referred to as target-device. The victim also owns a device with a microphone, hereafter referred to as recording-device. A genuine copy of some VoIP software is assumed to be installed on recording-device; this software is not compromised in any way. Also, recording-device is connected to the Internet and engaged in a VoIP call with at least one party who plays the role of the attacker. We underline that target-device and recording-device might not correspond — as long as target-device is close enough for recording-device microphone to pick up its keystroke sounds.

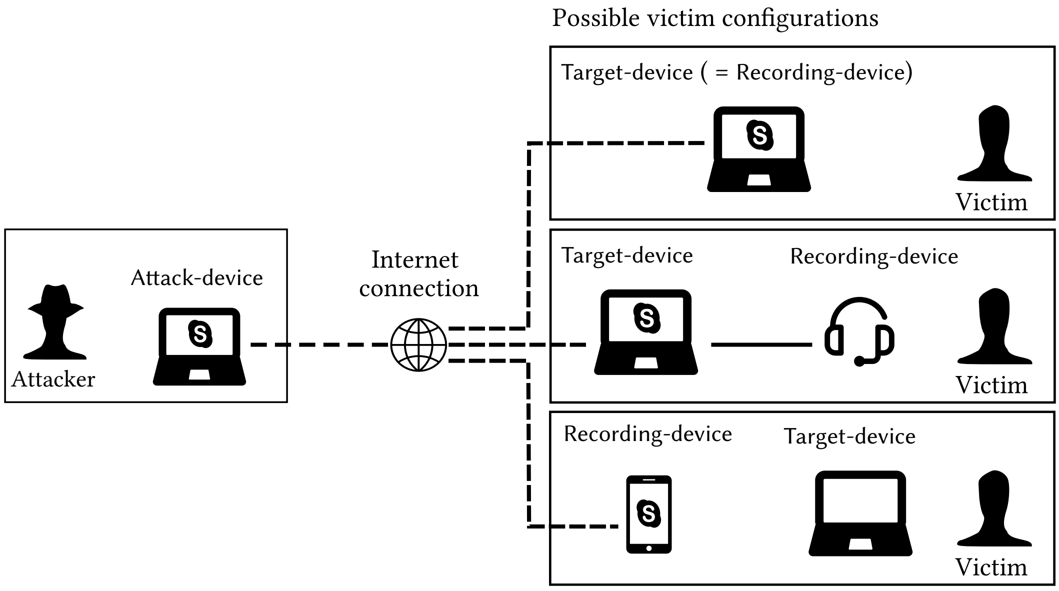


Fig. 1. System model. There are different possible configuration of target-device and recording-device. Victim's target-device needs not to correspond to recording-device — as long as there is physical proximity.

The attacker is a malicious user who aims to learn some private information about the victim. The attacker owns and fully controls a computer that we refer to as attack-device, which has a genuine (unmodified) version of the same VoIP software as recording-device. The attacker uses attack-device to receive and record the victim's acoustic emanations using VoIP software. We assume that the attacker relies solely on information provided by VoIP software. In other words, *during the attack*, the attacker receives no additional acoustic information from the victim, besides what VoIP software transmits to attack-device.

3.2 Threat Model

S&T attack transpires as follows: during a VoIP call between the victim and the attacker, the former types something on target-device, for example a text of an email message or a password. We refer to this typed information as target-text. Typing target-text causes acoustic emanations from target-device's keyboard, which are picked up by the recording-device's microphone and faithfully transmitted to attack-device by VoIP. The goal of the attacker is to learn target-text by taking advantage of these emanations.

We make the following assumptions:

- As mentioned above, the attacker has no real-time audio-related information beyond that provided by VoIP software. Acoustic information can be degraded by VoIP software by downsampling and mixing. In particular, without loss of generality, we assume that audio is converted into a single (mono) signal, as is actually the case with some VoIP software, such as Skype and Google Hangouts.

- If the victim discloses some keyboard acoustic emanations **together** with the corresponding plaintext — the actual pressed keys (called *ground truth*) — the volume of this information is small, on the order of a chat message or a short e-mail. We expect it to be no more than a few hundred characters.
- target-text is very short (e.g., $\approx$ 10 characters) and random, corresponding to an ideal password. This keeps *S&T attack* as general as possible, since dictionary words are a "special" case of random words, where optimization may be possible.

We now consider some realistic *S&T attack* scenarios. We describe them starting with the more generous setting where the attacker knows the victim's typing style and keyboard model, proceeding to the more challenging one where the attacker has neither type of information.

1) Complete Profiling: In this scenario, the attacker knows some of the victim's keyboard acoustic emanations on target-device, along with the ground truth for these emanations. This might happen if the victim unwittingly provides some text samples to the attacker during the VoIP call, e.g., sends chat messages, edits a shared document, or sends an email message[1]. We refer to such disclosed emanations as *"labeled data"*. To be realistic, the amount of labeled data should be limited to a few samples for each character.

We refer to this as *Complete Profiling* scenario, since the attacker has maximum information about the victim. It corresponds to attack scenarios used in prior supervised learning approaches [8, 23, 24, 33], with the difference that we collect acoustic emanations using VoIP software, while others collect emanations directly from microphones that are physically near target-device.

2) User Profiling: In this scenario, we assume that the attacker does not have any labeled data from the victim on target-device. However, the attacker can collect training data of the victim while the victim is using the same type of device (including the keyboard) as target-device[2]. This can be achieved via social engineering techniques or with the help of an accomplice. We refer to this as *User Profiling* scenario, since, unable to profile target-device, the attacker profiles the victim's typing style on the same device type.

3) Model Profiling: This is the most challenging, though the most realistic, scenario. The attacker has absolutely no training data for the victim. The attacker and the victim are engaged in a VoIP call and information that the attacker obtains is limited to the (unknown) victim keyboard's acoustic emanations.

The attacker's initial goal is to determine what laptop the victim is using. To do so, we assume that the attacker maintains a database of sounds from previous attacks. If the attacker already profiled the model of the current victim's target-device, it can use this information to mount the attack. We refer to this as *Model Profiling* scenario, since although the attacker can not profile the current victim, it can still profile a device of the same model as target-device. We observe that this scenario could also apply in the case where the attacker is not directly in the VoIP call, but compromised the VoIP software of the victim to act as a remote microphone.

4 SKYPE & TYPE ATTACK

S&T consists of two main activities: *training* attack models tailored on specific victims or keyboard models, and *classification* of unknown keystroke sounds — the actual attack. We divide *S&T* in 4

[1]Ground truth could also be collected offline, if the attacker happened to be near the victim, at some point before or after the actual attack. Note that this still does not require physical proximity between the attacker and the victim in *real time*.
[2]In case the target-device is a desktop, knowing the model of the desktop does not necessarily mean knowing the type of the keyboard. However, in mixed video/audio call the keyboard model might be visually determined, when the keyboard is placed in the visual range of the camera.

different phases, depicted in Figure 2 for the first three, and Figure 3 for the last phase. We first overview the four phases. Then, we detail all the steps of the four phases.

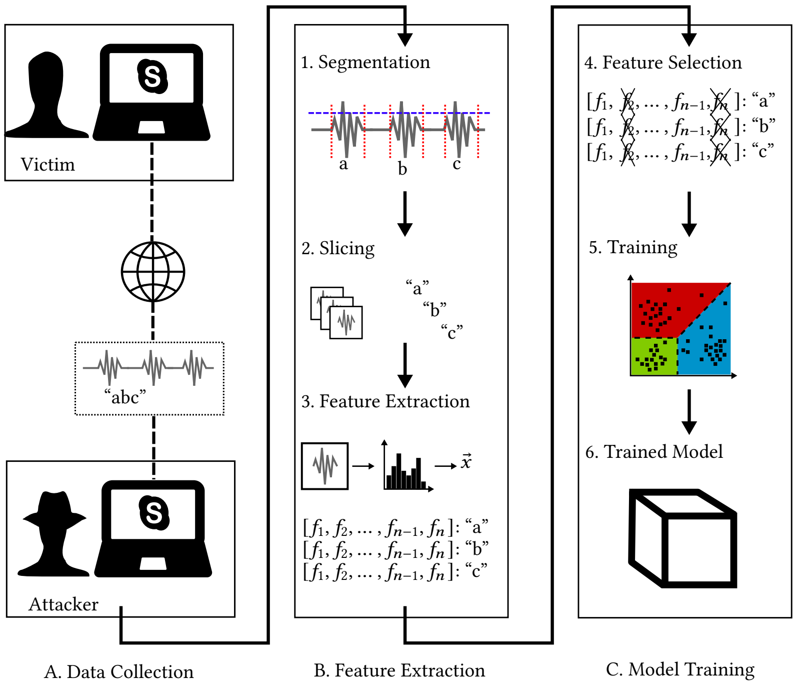

Victim
"abc"
Attacker
1. Segmentation
a
b
c
2. Slicing
"a"
"b"
"c"
3. Feature Extraction
$\vec{x}$
$[f_1, f_2, \dots, f_{n-1}, f_n]$: "a"
$[f_1, f_2, \dots, f_{n-1}, f_n]$: "b"
$[f_1, f_2, \dots, f_{n-1}, f_n]$: "c"
4. Feature Selection
$[f_1, \cancel{f_2}, \dots, f_{n-1}, \cancel{f_n}]$: "a"
$[f_1, \cancel{f_2}, \dots, f_{n-1}, \cancel{f_n}]$: "b"
$[f_1, \cancel{f_2}, \dots, f_{n-1}, \cancel{f_n}]$: "c"
5. Training
6. Trained Model
A. Data Collection
B. Feature Extraction
C. Model Training

Fig. 2. *S&T*, pre-processing steps: collection of victim's typing sounds (Step A), and machine learning pipeline to process them (Step B) and to train an attack model (Step C) if the keys corresponding to sounds are known.

A. **Data Collection -** The attacker receives keystroke sounds involuntarily leaked by the victim during a VoIP call. The attacker might know the keys corresponding to the sounds (and later use such sounds to train a machine learning model).
B. **Feature Extraction -** The attacker now pre-processes the keystroke sounds. He runs a pipeline that divides the recorded waveform, corresponding to all received keystrokes, into shorter waveforms, each containing the sound of a single keystroke. He then extracts meaningful features from the waveforms, obtaining one feature vector per keystroke.

After Phase B, the attacker can continue either with Phase C or Phase D. If the attacker knows the keys corresponding to each sound he received in Phase A, he continues with Phase C to train a tailored machine learning model:

C. **Model Training -** Using the labeled data, the attacker removes the least important features from the feature vectors (known as *feature selection*). Then, he trains a supervised machine learning model to classify keystroke sounds into different keys — and obtains a trained model that can be later used to perform *S&T attack*.

If the attacker during Phase A received non-labeled keystroke sounds, his goal is to infer the typed keys. Therefore, he continues to Phase D, where he uses a trained machine learning model to obtain the predictions:

D. **Attack Phase -** The exact steps depend on the specific threat model scenario. If the attacker already knows the attack model to use (i.e., the Complete Profiling and User Profiling scenarios),

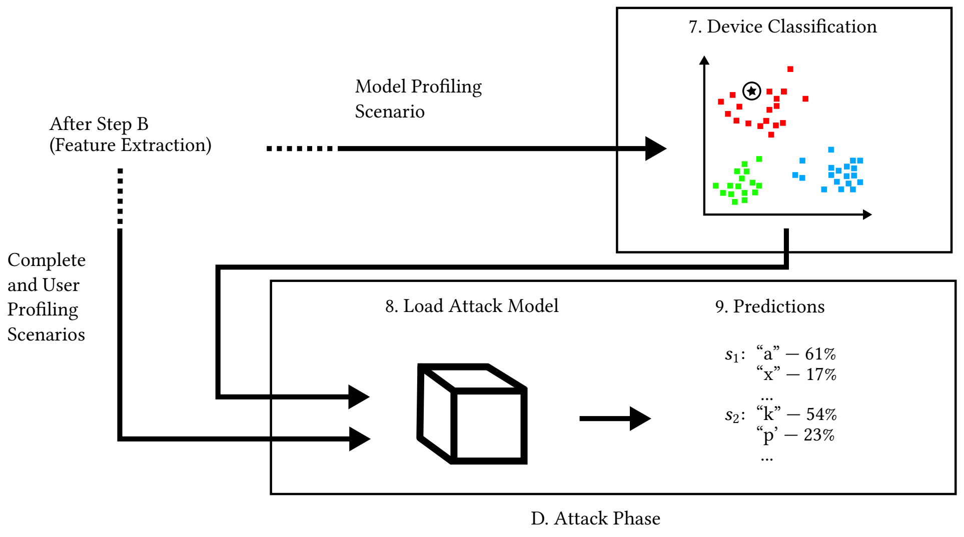


Fig. 3. *S&T*, attack steps: the attacker determines which attack model to load (depending on the scenario), and then predicts the keys of the given sounds (Step D).

he loads such model and uses it to predict the most likely keys for the keystroke sounds. Otherwise, in the challenging Model Profiling scenario, the attacker first performs Device-classification (to discover the likely keyboard in use by the victim), then loads the indicated model and proceeds to obtain predictions.

We now describe in more details all four phases and their different steps. Throughout the rest of the section, we refer to the phases and steps presented in Figure 2 and Figure 3.

4.1 Phase A: Data Collection

All envisaged scenarios involve the attacker engaged in a VoIP call with the victim. During the call, the victim types something on target-device's keyboard. As expected, the VoIP software records and transmits the keystroke sounds to the attacker. The attacker then records such sounds — for example by channeling VoIP output to some local recording software. If the attacker knows the keys corresponding to the keystroke sounds (for example, as discussed in Section 3.2, thanks to a common chat, shared document, or email), he keeps these annotations to generate *labeled data*. The attack then proceeds to the next phase.

4.2 Phase B: Feature Extraction

The main goal in this phase is to extract meaningful features from acoustic information. The first step is segmentation, needed to isolate distinct keystroke sounds within the recording. The attacker then slices the waveform into sound samples, containing one keystroke sound each. Subsequently, using these sound samples, we build derived values (called features) that represent properties of the keystroke sounds. This step is referred to as feature extraction.

The details of the steps are the following:

1. **Segmentation.** We perform data segmentation according to the following observation: the waveform of a keystroke sound presents two distinct peaks, shown in Figure 4. These two peaks correspond to the events of: (1) the finger pressing the key — *press* peak, and (2) the finger

releasing the key — *release* peak. Similar to [8], we only use the press peak to segment the data and ignore the release peak. This is because the former is generally louder than the latter and is thus easier to isolate, even in very noisy scenarios.

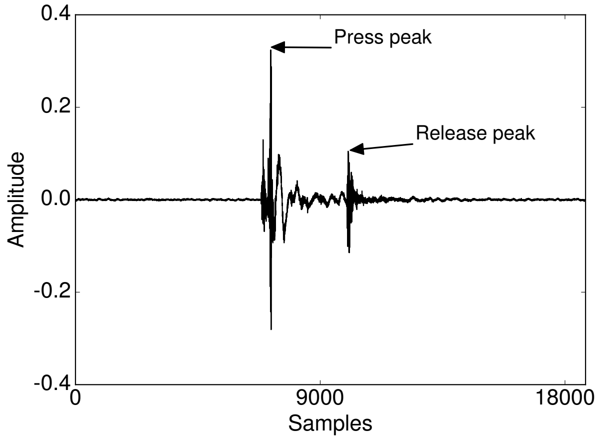


Fig. 4. Waveform of the "*A*" key, recorded on an Apple Macbook Pro 13" laptop.

To perform automatic isolation of keystrokes, we set up a detection mechanism as follows: we first normalize the amplitude of the signal to have root mean square of 1. We then sum up the FFT coefficients over small windows of 10ms, to obtain the energy of each window. We detect a press event when the energy of a window is above a certain threshold. The threshold is a parameter that can be tuned. We observed that background noise is usually significantly quieter than keystroke sounds — therefore, all points in time when the intensity of the waveform is above the 90th percentile of all intensities are most likely "events" where, e.g., a keystroke happened. In case of more background noise, or external loud noises, the threshold might need manual tuning by the attacker, or manual data cleaning to remove the spurious high-intensity sounds (e.g., a door slamming). The same holds if there is overlap of voice and keystroke sounds — the attacker might need to manually increase the threshold in order to only capture the high-intensity keystroke sounds.

2. **Slicing.** For each event detected by the data segmentation step, we then extract the subsequent 100ms [12, 45] of waveform as the keystroke event waveform. If keystroke sounds are very close to each other (i.e., less than 100ms), it is possible to extract a shorter waveform with minimal loss in accuracy (see Section 5.1).
3. **Feature Extraction.** As features, we extract the mel-frequency cepstral coefficients (MFCC) [30]. These features capture statistical properties of the sound spectrum, which is the only information that we can use. Indeed, due to the mono acoustic information, it is impossible to set up an attack that requires stereo audio and uses TDoA, such as [29, 40, 44]. Among possible statistical properties of the sound spectrum — including: MFCC, FFT coefficients, and cepstral coefficients — we chose MFCC which yielded the best results. To select the most suitable property as a feature, we ran the following experiment:

   Using a Logistic Regression classifier we classified a dataset with 10 samples for each of the 26 keys corresponding to the letters of the English alphabet, in a 10-fold cross-validation scheme. We then evaluated the accuracy of the classifier with various spectral features: FFT coefficients, cepstral coefficients, and MFCC.

We repeated this experiment with data from five users on a Macbook Pro laptop. Accuracy results were as follows: 90.61% (± 3.55%) for MFCC, 86.30% (± 6.34%) for FFT coefficients, and 51% (± 18.15%) for cepstral coefficients. This shows that MFCC offers the best features. For MFCC experiments we used parameters similar to those in [45]: a sliding window of 10ms with a step size of 2.5ms, 32 filters in the mel scale filterbank, and used the first 32 MFCC.

4.3 Phase C: Model Training

This phase happens when the attacker received keystroke sounds as labeled data, i.e., together with the corresponding keys. With labeled data, the attacker can train a supervised learning classification model. We argue that the attacker needs to have labeled data, and use a supervised classification technique: as discussed in Section 3.2, approaches that require lots of data to cluster, such as [12], are incompatible with our assumptions, because we might have only a small amount of both training and testing data. Moreover, potential randomness of target-text makes it impossible to realize constraint-based approaches, which would require target-text to be a meaningful word, as in [45].

The detailed steps of this phase are the following:

4. **Feature Selection.** The attacker determines the most relevant features, among all features produced by the Feature Extraction step. To do so, *S&T* uses a Recursive Feature Elimination algorithm [22], that tries to find the best performing subset of features among the full set. This step helps eliminating features that are irrelevant and only capture background noise or non-discriminating parts of the sound spectrum.
5. **Training.** The attacker now needs to train a machine learning model to discriminate between different keys based on their keystroke sounds. We consider key classification to be a multiclass classification problem, where different classes correspond to different keyboard keys. Therefore, the input to the classifier for training is a set of feature vectors, representing properties of keystroke sounds — each associated to the specific key that generated the sound. Ideally, the attacker should have more than one sample for each key. The classifier then learns to discriminate between different keys of target-device by sound properties.
   To perform key classification, we use a Logistic Regression (LR) classifier, since it outperformed other tested classifiers, including: Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), Random Forest (RF), and k-nearest neighbors. We tested this in a preliminary experiment that uses each candidate to classify a dataset of 10 samples, for each of the 26 keys corresponding to the letters of the English alphabet, in a 10-fold cross-validation scenario. We use MFCC as features, and, for each classifier, we optimized its hyper-parameters with an extensive grid search: we repeated training and testing with all possible combinations of hyper-parameters, and selected the best performing combination.
   To evaluate the classifiers' quality in this experiment, we used *accuracy* and *top-n accuracy* measures. Given true values of k, accuracy is defined in the multiclass classification case as the fraction of correctly classified samples over all samples. Top-n accuracy is defined similarly. The sample is correctly classified if it is present among the top n guesses of the classifier. Results of this preliminary experiment are shown in Figure 5 which demonstrates that the best performing classifiers are LR and SVM. This is especially the case if the classifier is allowed to make a small number of predictions (between 1 and 5), which is more realistic in an eavesdropping setting. In particular, both LR and SVM exhibit around 90% top-1 accuracy, and over 98.9% top-5 accuracy. However, LR slightly outperforms SVM until top-4 — for this reason, we selected LR for *S&T*.

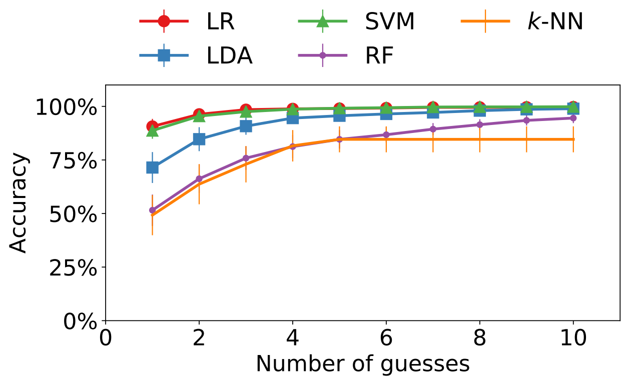


Fig. 5. Average top-n accuracy of single key classification, as a function of the number of guesses, for each of the tested classifiers.

6. **Trained Model.** Once the attacker trained his LR model with the labeled data, he stores such model and any important auxiliary information (e.g., the features to eliminate in future samples). This model is tailored to the specific scenario the attacker trained his model for.

4.4 Phase D: Attack Phase

This phase happens when the attacker receives some keystroke sounds without knowing their corresponding keys. For example, the victim might have typed his password on some website during the VoIP call. In this case, the attacker wants to predict the keys corresponding to the received sounds. The exact steps depend on the threat scenario, as shown in Figure 3: in Complete and User Profiling scenarios, the attacker knows which attack model to load, and can directly proceed to Step 8. Instead, in Model Profiling scenarios, the attacker does not know the target-device used by the victim — he therefore needs to first understand this information by performing Step 7, Target-device classification.

7. **Device-classification.** In *Model Profiling* scenario, since the attacker has no knowledge of the victim's typing style or target-device, it begins by trying to identify target-device by classifying its keyboard sounds. We consider the task of target-device classification as a multiclass classification problem, where different classes correspond to different target-device models known to the attacker. In particular, the attacker maintains a database of keystroke sounds corresponding to many different keyboard models, and wishes to predict which keyboard model generated some other unknown keystroke sounds.
   To perform this classification task, we use a k-nearest neighbors (k-NN) classifier with $k = 10$ neighbors, that outperformed other classifiers such as Random Forest and Logistic Regression in our preliminary experiments. The k-NN classifier, when given an unknown sample (i.e., the feature vector of a keystroke sound) finds the k closest samples in terms of sound features, and outputs their corresponding target-device. The prediction for the target-device that generated such sound is the majority of these outputs.
   The output of this phase is the keyboard used by the victim (i.e., his target-device), or that the target-device is unknown. In this latter case, the attacker can not perform *S&T*.
8. **Load Attack Model.** Once the attacker knows the target-device used by the victim, and depending on the specific scenario, he loads the specific model previously trained on the appropriate data. This model, given a feature vector corresponding to a keystroke sound, retains only the

appropriate features that were determined by Step 4 in its previous training, and then generates predictions for the keystroke.

9. **Predictions.** Given a feature vector of a waveform of a keystroke, attack models can return a list of predictions of probable keys corresponding to the pressed key that generated the sound. Such predictions are ordered by probability, such that, e.g., the first 5 predictions returned by the model correspond to the 5 most likely keys according to the classifier. These ordered predictions help build further use-cases for *S&T*, for example prediction of typed words (Section 7.1) or of passwords (Section 7.2).

5 EXPERIMENTAL SETUP

To assess the feasibility of *S&T attack* on all considered scenarios and combinations of target-devices and recording-devices described in Section 3, we collected a large dataset of keystroke sounds on a variety of settings. In this section, we first describe the two data collection campaigns that we carried out, and motivate them by illustrating the different hypotheses we planned to test on each campaign in Section 5.1. We then discuss the demographics of our study participants, the implications and generality of our study, and detail the experiment protocol in Section 5.2.

5.1 Data Collection

We performed two separate data collection campaigns:

- **First Campaign.** We collected data to test our different attack scenarios (Complete, User, and Model Profiling, introduced in Section 4) — that requires to have multiple target-devices of the same model — and the difference between typing styles, i.e., *Hunt and Peck* and *Touch* typing (to simulate typists of different skill levels).
- **Second Campaign.** Besides expanding our collection of target-devices, we collected data to test the impact of different recording-devices on *S&T*. Moreover, we tested the impact of typing speed by requiring participants to type a popular sentence in English — a natural activity, that allowed participants to type at their own pace.

We summarize our broad choice of devices and setups in Table 1. In total, we collected data from 12 unique participants and 14 keyboards of 11 different models. Considering the use of different recording-devices, we collected 156 unique datasets.

First Campaign. We collected data from 5 distinct users. For each user, the task was to press the keys corresponding to the English alphabet, sequentially from “A” to “Z”, and to repeat the sequence ten times, first by only using the right index finger (this is known as *Hunt and Peck* typing, referred to as *HP* from here on), and then by using all fingers of both hands (*Touch* typing) [24]. We believe that typing letters in the order of the English alphabet rather than, for example, typing English words, did not introduce bias. Typing the English alphabet in order is similar to typing random text, that *S&T attack* targets. Moreover, a very fast touch typist usually takes around 80ms to type consecutive letters [14]. We show that this is the case with our data in Section 6.4; moreover, we argue that *S&T attack* works without any accuracy loss with samples shorter than this interval. In order to test correctness of this assumption, we ran a preliminary experiment as follows:

> We recorded keystroke audio of a single user on a Macbook Pro laptop typing the English alphabet sequentially from “A” to “Z” via Touch typing. We then performed Step B (Feature Extraction) and Step C (Model Training) as described in Section 4. However, instead of extracting 100ms of the waveform, we extracted 3ms (as in [8]), and then from 10ms to 100ms at intervals of 10ms for each step. We then tested *S&T attack* in a 10-fold cross-validation scheme. Figure 6 shows top-5 accuracy of this preliminary experiment, for different lengths of the sound sample that we extracted.

Table 1. Configuration details of our data collection campaigns.

	First Setup	**Second Setup**
Users	5	8
Target Devices	6 *(2 laptops of 3 different models)*	4 laptops 3 USB membrane keyboard 1 USB mechanical keyboard
Typing Style	Hunt&Peck Touch typing	Touch typing
Recording Devices	Laptop microphone	Laptop microphone Smartphone Headset microphone
Typed Text	Ten times the entire English alphabet (a-z), in order	Ten times the sentence “The quick brown fox jumped over the lazy dog”
Total Datasets	60 *(5 users · 6 considered keyboards · 2 typing styles)*	96 *(4 users · 4 considered keyboards · 3 recording devices) — repeated twice*

We observe that, even with very short 20ms samples, *S&T attack* suffers minimal accuracy loss. Therefore, we believe that adjacent letters do not influence each other, since sound overlapping is very unlikely to occur.

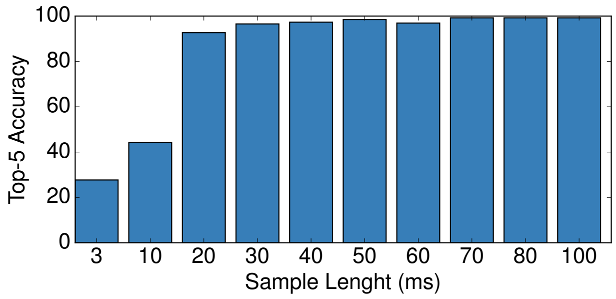


Fig. 6. Top-5 accuracy of single key classification for different sample lengths.

Note that collecting only the sounds corresponding to letter keys, instead of those for the entire keyboard, does not affect our experiment. The “acoustic fingerprint” of every key is related to its position on the keyboard plate [8]. Therefore, all keys behave, and are detectable, in the same way [8]. Due to this property, we believe that considering only letters is sufficient to prove our point. Moreover, because of this property, it would be trivial to extend our approach to various keyboard layouts, by associating the keystroke sound with the position of the key, rather than the symbol of the key, and then mapping the positions to different keyboard layouts.

Every user performed the data collection task on six laptops: (1) two Apple Macbook Pro 13" 2014, (2) two Lenovo Thinkpad E540, and (3) two Toshiba Tecra M2. We selected these as being representative of many common modern laptop models: Macbook Pro is a very popular aluminium-case high-end laptop, Lenovo Thinkpad E540 is a 15" mid-priced laptop, and Toshiba Tecra M2 is an older laptop model, manufactured in 2004. We needed pairs of laptops of the same model to test the User Profiling and Model Profiling scenarios. All acoustic emanations of the laptop keyboards were recorded by the microphone of the laptop in use, with Audacity software v2.0.0. We recorded all data with a sampling frequency of 44.1kHz, and then saved it in WAV format, 32-bit PCM signed.

Second Campaign. We collected data from 8 distinct users — 7 of which did not participate in the first data collection campaign. The task for each user was to type a famous pangram (i.e., a sentence with all the letters of English alphabet): *"The quick brown fox jumps over the lazy dog"*, as it is short and simple. All users typed the sentence ten times for each considered keyboard, using all fingers of both hands (Touch typing).

We recorded these users on 8 different keyboards: 4 laptop keyboards, and 4 external USB keyboards. We selected the devices to be representative of many common user setups. Regarding laptops, we tested an HP 250 G5 (a low-priced domestic laptop), a Dell Inspiron 13 5000 (mid-range business laptop), and a Dell XPS 13 9333 and an Asus Zenbook UX303UB (high-end slim laptops with aluminium bodies). Regarding external keyboards, we tested 3 popular office membrane keyboards — HP 9009, Fujitsu KB900, HP KU-0316 — and a mechanical keyboard, a AUKEY KM-G3 equipped with "OTEMU Blue" mechanical switches, with a distinctive "clicking" sound typical of mechanical switches.

We concurrently recorded keystroke sounds with three different recording devices:

- The **laptop** in use — external USB keyboards were recorded using either the Dell XPS 9333 laptop microphone (HP 9009, AUKEY KM-G3) or the Dell Inspiron 5000 laptop microphone (Fujitsu KB900, HP KU-0316);
- A **smartphone** placed in proximity of the laptop/keyboard in use — either a Samsung Galaxy S6 (HP 9009, AUKEY KM-G3) or a Huawei P10 lite (Fujitsu KB900, HP KU-0316);
- A **headset** with a microphone, worn by the study participant — we used a generic Samsung inexpensive headset, connected to a Huawei P9 Lite.

All recordings were done with a sampling frequency of 44.1kHz, and then saved in WAV format, 32-bit PCM signed.

Resulting Datasets. All recorded data was then processed through a VoIP software — we chose Skype as representative, as (i) it is one of the most popular VoIP tools [3–5]; (ii) its codecs are used in Opus, an IETF standard [38][3]; (iii) it reflects our general assumption about mono audio.

We filtered the sound samples by routing the recorded sounds through the Skype software, and recording the received emanations on a different computer (i.e., on the attacker's side). As we saved raw data from the microphones at their native quality, there is no difference between immediately recording Skype-filtered data, or filtering at a later moment, as we did: Skype receives the same data as if it were receiving it from the microphone in the first place. To perform this filtering, we used two machines running Linux, with Skype v4.3.0.3 for the first setup, and Skype v8.11.0.4 and 8.34.0.78 for the second setup[4], connected via a high-speed network. During the calls, there was no sensible data loss. We analyze bandwidth requirements needed for data loss to occur, and the impact of bandwidth reduction, in Section 6.4.

[3]Opus is employed in many other VoIP applications, such as Google Hangouts and Teamspeak [1].
[4]Between the first and second data collection campaigns, Skype v4 was discontinued by Microsoft.

At the end of data collection and processing phases we obtained datasets for the first and second setups. For the first set of experiments, each dataset consists of 260 samples, 10 for each of the 26 letters of the English alphabet. For the second set of experiments, each dataset consists of 10 repetition of the aforementioned pangram.

In total, we recorded 12 unique participants and 14 keyboards of 11 different models. As mentioned, considering the use of different recording-devices, we collected 156 unique datasets. The number of users and of keyboards we considered greatly exceeds related works on keyboard acoustic eavesdropping [8, 23, 24, 33]: such works evaluated 1–2 users on 1–3 keyboards, testing on a quantity of data comparable to only 10 of our datasets. We further discuss generality in the next section.

5.2 Study Participants

We recruited a total of 12 unique participants for our study. Participants were randomly selected among (primarily) college and graduate students who agreed to be part of the experiment. Five participated in the first data collection campaign, and eight in the second campaign — one participant joined both the first and the second campaign. Out of 12, 4 were female and 8 were male. The youngest was 20 years old at the time of the experiment, while the oldest was 31; the mean age was 25.8 (±3.07). Nine participants had a Computer Science/Engineering background, two had humanities backgrounds, one had a high school diploma. All participants self-reported to be confident with typing on a computer keyboard, and experienced in the use of computers. All of them already knew how to type with both hands (Touch typing), with different degrees of confidence.

The participants' typing speed was diverse. We recorded the inter-key time of the 8 participants to the second data collection campaign (where the task was to type an English sentence), and show them in Figure 7.

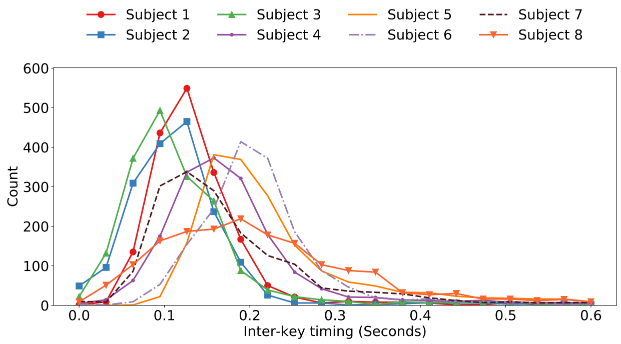


Fig. 7. Distribution of inter-key timing of experiment participants.

We can see that three participants are consistently fast typists (subjects 1, 2, and 3), three are somewhat average-speed typists (subjects 4, 5, 6), one is a slow typist (Subject 7), and one is noticeably non-uniform in typing speed (Subject 8). There is no apparent correlation between demographics and average speed: one of the fast typists has a background in humanities, while the slowest is in Computer Science.

We believe that typing speed is a good proxy for typing experience, therefore our diverse population can be representative of skilled and less skilled typists. Moreover, less skilled typists are simulated by our HP typing style. Furthermore, intuitively skilled typists represent the worst case for *S&T*, because of the increased probability of overlapping keystroke sounds. However, our population exhibits an age bias, skewed towards young people, and is thus unclear how it generalizes to greatly different age groups.

Experimental Protocol. Approval from an Institutional Review Board was not mandatory to collect and handle the recorded data in the institution where the tests were carried out. However, we followed well-established good practices in data collection and handling, that we now report in detail.

Participants were asked to sit on a chair in front of a computer with a keyboard. Where applicable, they were asked to wear a headset microphone. They got time to relax and get acquainted with the task, whether it was about typing the English alphabet or the pangram sentence. Participants were only required to type in a specific way when testing the HP typing style hypothesis, otherwise they were encouraged to type in their usual way, with as many fingers as desired. Participants (who signed an informed consent) were informed that they were free to interrupt the experiment at any time and request deletion of their data if they desired. There was no incentive for participants to complete the task as fast as possible, or to meet any performance goal (e.g., participants were not penalized for making mistakes, and did not have to re-type a sentence in case of mistakes) — therefore, we believe there was no bias in our data collection procedure.

Data was handled confidentially, and is stored encrypted in our laboratory. The data is anonymized together with the identities of the participants, and was only used for the specified purpose of running this study. The mapping between anonymous identifiers and demographic information is securely stored in a different location.

6 S&T ATTACK EVALUATION

We evaluated *S&T attack* in all scenarios described in Section 3.2:

- We report *Complete Profiling* scenario in full detail (Section 6.1), by analyzing performance of *S&T attack* separately on all considered keyboards, typing styles, and VoIP filtered and unfiltered data. These results serve as a reference for the accuracy of *S&T attack* on other scenarios.
- We then analyze *User Profiling* (Section 6.2) and *Model Profiling* (Section 6.3) scenarios only on realistic settings, namely Touch typing, VoIP filtered data. We compare these results with results of *Complete Profiling*, to understand the impact on accuracy of less powerful adversaries.
- We thoroughly evaluate the impact of different conditions on the baseline *Complete Profiling* scenario (Section 6.4). We explore the impact on *S&T attack* accuracy of different typists, different inexpensive recording devices that are commonly used in VoIP calls, small training sets, and VoIP-specific issues such as degradation of call quality due to low bandwidth, and voice on top of keystroke sounds.
- We conclude by discussing our results and their significance (Section 6.5).

Evaluation Technique. Unless otherwise specified, we evaluate *S&T* accuracy on a given dataset as follows. We consider the given dataset in a stratified 10-fold cross-validation scheme[5]. For every fold, we proceed with Step C and Step D of *S&T* (see Section 4): we perform feature selection on training data using Recursive Feature Elimination [22], and then train a machine learning model. We then calculated the accuracy of the classifier over each fold, and then computed the mean and standard deviation of accuracy values.

[5]In a stratified k-fold cross-validation scheme, the dataset is split in k sub-samples of equal size, each having the same percentage of samples for every class as the complete dataset. One sub-sample is used as testing data, and the other $(k-1)$ — as training data. The process is repeated k times, using each of the sub-samples as testing data.

We underline that data collected in our first campaign is already balanced (i.e., there are 10 samples for each letter), while data collected in our second campaign is not (i.e., we have 10 repetitions of the pangram, where letters occur with different frequencies). Therefore, every time we use data from the second campaign, we perform *undersampling*: we randomly select a subset of the data such that every letter appears 10 times — and thus the resulting dataset is balanced. To mitigate randomness in the undersampling process, we repeat it for 5 times, and then average the results.

Baseline. We evaluated the accuracy of *S&T attack* in recognizing single characters, according to the top-n accuracy, defined in [13]. As a baseline, we considered a random guess with accuracy n/l, where n is the number of guesses, and l is the size of the alphabet. Therefore, in our experimental setup, accuracy of the random guess is $n/26, n \in [1 \dots 26]$, since we considered 26 letters of the English alphabet. Because of the need to eavesdrop on random text, we can not use "smarter" random guesses that, for example, take into account letter frequencies in a given language. Moreover, these results serve as a baseline to more complex attacks that consider additional information (e.g., the language of target-text).

6.1 Complete Profiling Scenario

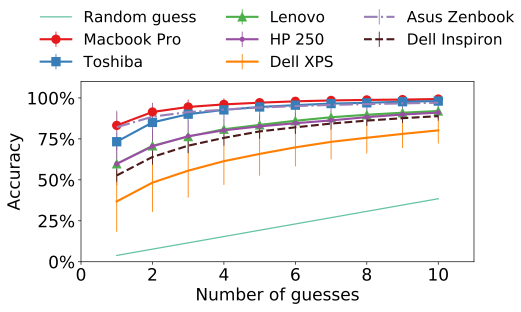
To evaluate the scenario where the victim disclosed some labeled data to the attacker, we considered all datasets registered with laptop internal microphones (i.e., all datasets of the first setup, and a subset of the datasets of the second), one at a time, following our evaluation technique.

Figure 8 depicts results of the experiment on the realistic Touch typing, Skype-filtered data combination. We observe that *S&T attack* achieves its lowest performance on the Dell XPS laptop and on the Fujitsu USB keyboard (recorded through the Dell Inspiron 5000 internal microphone), with low top-1 accuracies of 36.81% and 38.92%, but satisfactory top-5 accuracies of 65.81% and 68.02%, respectively. Performance on the other USB keyboards are similar to the Fujitsu, with the mechanical AUKEY keyboard being the best performing at a top-1 accuracy of 48.08% and top-5 of 79.72%. Interestingly, the second worst performing laptop is the Dell Inspiron 5000 with top-1 and top-5 accuracies of 52.67% and 79.56% — given that the four USB keyboards were recorded using the two Dell laptops' integrated microphones, these results hint to these microphones leading to inaccurate recordings.

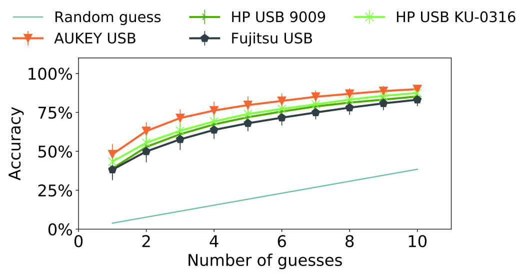
Lenovo laptops have intermediate performance, with top-1 accuracy of 59.8%, and a top-5 accuracy 83.5%. The similar HP laptop exhibits similar performance. On the Macbook Pro, Asus Zenbook, and Toshiba, we obtained very high top-1 accuracy between 73.3% and 83.23%, and top-5 accuracy between 94.5% and 97.1%. We believe that these differences are due to variable quality of manufacturing, e.g., the bodies and keyboards of our particular Lenovo and HP laptops are made of cheap plastic materials. Another possible reason are different microphones: we show in Section 6.4 that different recording devices can have an impact on accuracy up to 10%.

Confusion Matrices. We extend our investigation to classification *mistakes*: we analyzed the confusion matrices of our classifiers, to see if misclassifications give us some insights. We report the confusion matrices for two sample keyboards, the Asus Zenbook and Dell Inspiron 5000 laptops, in Figure 9 (confusion matrices for other keyboards look similar and exhibit similar patterns).

We observe that many misclassifications involve *neighboring* keys on the keyboard: some examples are y–u and t–y on the Asus Zenbook, x–c, and s–w on the Dell Inspiron, and c–f, q–w, t–g, and w–e on both keyboards. These results further confirm the observations of Asonov and Agrawal [8] regarding the influence of position on keyboards on keys' sound — and suggest that it is possible to further increase the accuracy of *S&T* by biasing the classifiers' predictions to exploit locality.

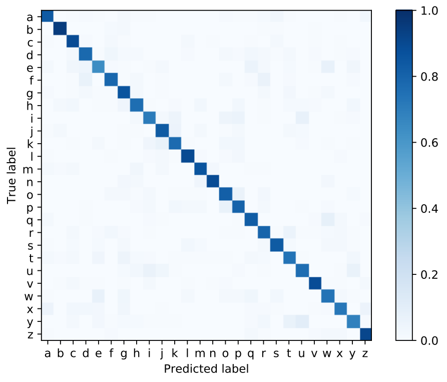


(a) Laptop Keyboards.

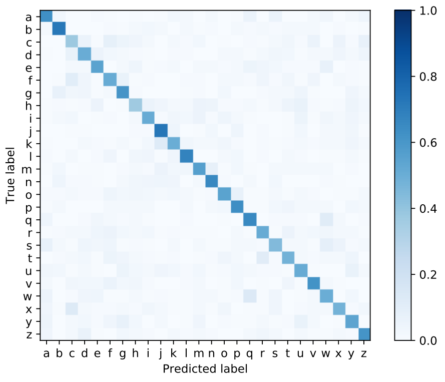


(b) USB Keyboards.

Fig. 8. *S&T attack* performance — *Complete Profiling* scenario, Touch typing, Skype-filtered data, average accuracy.



(a) Asus Zenbook.



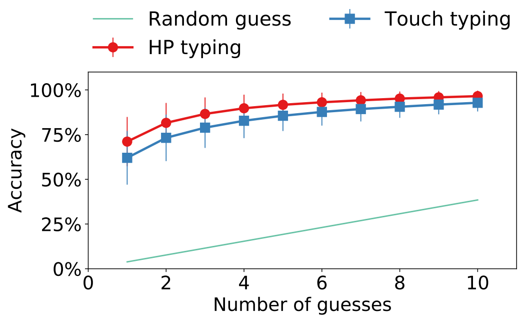
(b) Dell Inspiron.

Fig. 9. *S&T attack* performance — confusion matrices.

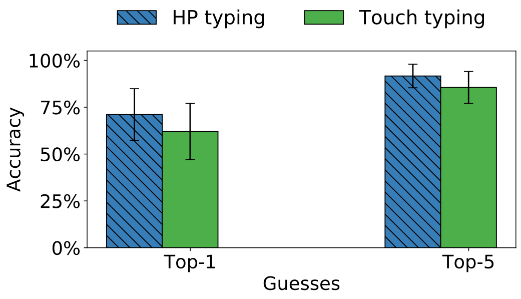
Impact of Typing Style and VoIP Software. Interestingly, we found that there is little difference between this data combination (*Touch* typing, Skype-filtered data, that we consider the most unfavorable) and the others. In particular, we compared average accuracy of *S&T attack* on HP and Touch typing data, and found that the average difference in accuracy is 0.80%. Such results are reported in more detail in Figure 10. Moreover, we compared the results of unfiltered data with Skype filtered data, and found that the average difference in accuracy is a surprising 0.33%. This clearly shows that Skype does not reduce accuracy of *S&T attack*.

Generality of VoIP Software. We also ran a smaller set of these experiments over Google Hangouts and observed the same tendency, that we show in Figure 11. This means that the keyboard acoustic eavesdropping attack is applicable to other VoIP software, not only Skype. It also makes this attack more credible as a real threat.

From now on, we only focus on the most realistic combination — Touch typing and Skype filtered data. We consider this combination to be the most realistic, because *S&T attack* is conducted over Skype, and it is more common for users to type with the Touch typing style, rather than the HP

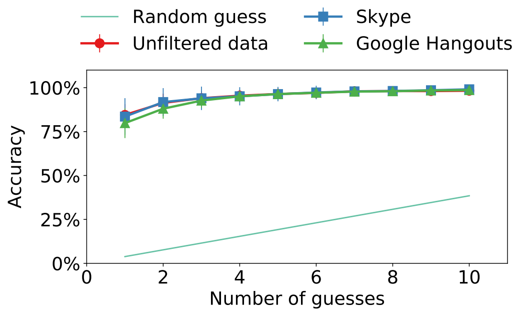


(a) Average accuracy.

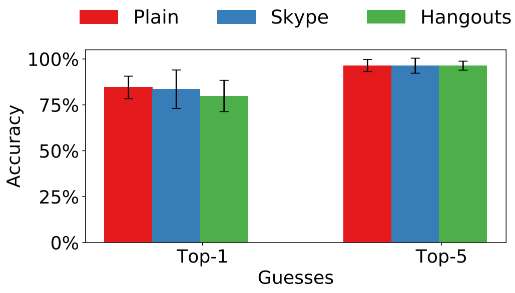


(b) Top-1 and top-5 accuracy.

Fig. 10. *S&T attack* performance — accuracy of HP and Touch typing data.



(a) Average accuracy.



(b) Top-1 and top-5 accuracy.

Fig. 11. *S&T attack* performance — accuracy of unfiltered, Skype-filtered and Google Hangouts-filtered data.

typing style. We limit ourselves to this combination to further understand real-world performance of *S&T*.

6.2 User Profiling Scenario

In this case, the attacker profiles the victim on a laptop of the same model of target-device. We selected the dataset of a particular user on one of the six laptops, and used it as our training set. Recall that it includes 260 samples, 10 for every letter. This training set modeled data that the attacker acquired, e.g., via social engineering techniques. We used the dataset of the same user on the other laptop of the same type, to model target-device. We conducted this experiment for all six laptops.

Results reflected in Figure 12 show that top-1 accuracy decreases to as low as 14% on Toshiba and Lenovo laptops, and to 19% on Macbook Pro. However, top-5 accuracy grows to 41.9%, 54%, and 45.6% on Lenovo, Macbook Pro, and Toshiba, respectively. This shows the utility of social engineering techniques used to obtain labeled data of the victim, even on a different laptop.

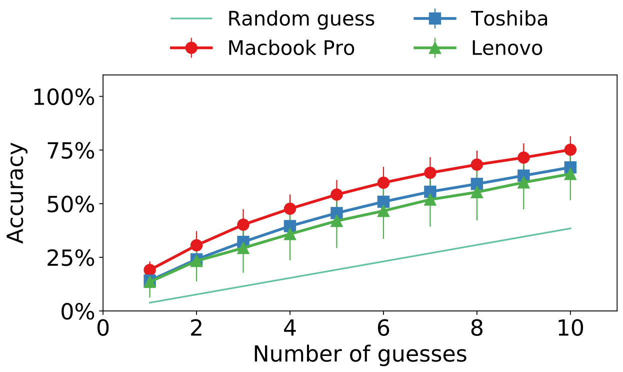


Fig. 12. *S&T attack* performance — *User Profiling* scenario, average accuracy.

6.3 Model Profiling Scenario

We now evaluate the most unfavorable and the most realistic scenario where the attacker does not know anything about the victim. Conducting *S&T attack* in this scenario requires: (i) target-device classification, followed by (ii) key classification.

Target-device classification. The first step for the attacker is to determine whether target-device is a known model. We assume that the attacker collected a database of acoustic emanations from many keyboards.

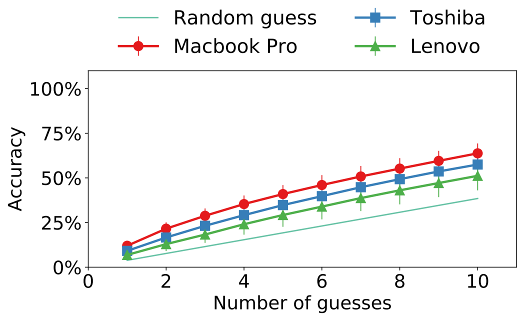
When acoustic emanations from target-device are received, if the model of target-device is present in the database, the attacker can use this data to train the classifier. To evaluate this scenario, we completely excluded all records of one user and of one specific laptop of the original dataset. We did this to create a training set where both the victim's typing style and the victim's target-device are unknown to the attacker. We also added to the training set several devices, including 7 USB keyboards (Apple Pro, Logitech Internet, Logitech Y, and the four USB keyboards we recorded for our experiments), as well as 6 laptops (Acer E15, Sony Vaio Pro 2013, and the four laptops we recorded in the second data collection campaign: HP 250, Dell XPS, Asus Zenbook, and Dell Inspiron). We did this to show that a laptop is recognizable from its keyboard acoustic emanations among many different models: our experiments try to recognize one target-device model among 16 different candidate devices.

We evaluated the accuracy of k-NN classifier in identifying the correct laptop model, on the Touch typing and Skype filtered data combination. Results show very high accuracy of 96%, while a random guess would only have a probability of succeeding of $1/16 = 6.25\%$. This experiment confirms that an attacker can determine the victim's device, by using acoustic emanations.

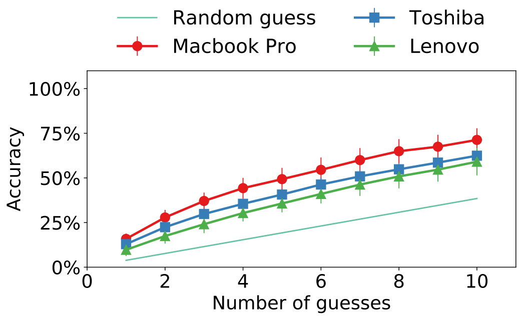
We now consider the case when the model of target-device is not in the database. The attacker must first determine that this is indeed so. This can be done using the confidence of the classifier. If target-device is in the database, most samples are classified correctly, i.e., they "vote"correctly. However, when target-device is not in the database, predicted labels for the samples are spread among known models. One way to assess whether this is the case is to calculate the difference between the mean and the most-voted labels. We observed that trying to classify an unknown laptop consistently leads to a lower value of this metric: 0.22 *vs* 0.49. The attacker can use such observations, and then attempt to obtain further information via social engineering techniques, e.g., laptop [27], microphone [18] or webcam [31] fingerprinting.

Key classification. Once the attacker learns target-device, it proceeds to determine keyboard input. However, it does not have any extra information about the victim that can be used to train the classifier. Nonetheless, the attacker can use, as a training set, data obtained from another user on a laptop of the same model as target-device.

Results of *S&T attack* in this scenario are shown in Figure 13a. As expected, accuracy decreases with respect to previous scenarios. However, especially with Macbook Pro and Toshiba datasets, we still have an appreciable advantage from a random guess baseline. In particular, top-1 accuracy goes from a 178% improvement from the baseline random guess on Lenovo datasets, to a 312% improvement on Macbook Pro datasets. Top-5 accuracy goes from a 152% on Lenovo to a 213% on Macbook Pro.



(a) *Model Profiling* scenario



(b) *Model Profiling* scenario, "crowd" training data

Fig. 13. Average attack accuracy

To further improve these results, the attacker can use an alternative strategy to build the training set. Suppose that the attacker recorded multiple users on a laptop of the same model of the target-device and then combines them to form a "crowd" training set. We evaluated this scenario as follows. We selected the dataset of one user on a given laptop, as a test set. We then created the training set by combining the data of other users of the same laptop model. We repeated this experiment, selecting every combination of user and laptop as a test set, and the corresponding other users and laptop as a training set. Results reported in Figure 13b show that overall accuracy grows by 6-10%, meaning that this technique further improves classifier's detection rate. In particular, this increase in accuracy, from 185% to 412% (with respect to a baseline random guess) yields a greater improvement than the approach with a single user on the training set.

Results show that *S&T attack* is still quite viable in a realistic VoIP scenario, with a target text which is both short and random. Moreover, this is possible with little to none specific training data of the victim, i.e., the attacker might even have *no prior knowledge* of the victim.

6.4 Variations

We thoroughly investigated the impact of different conditions on the baseline *Complete Profiling* scenario. In the following, we report the impact on *S&T attack* accuracy of different typists, different inexpensive recording devices that are commonly used in VoIP calls, small training sets, and of VoIP-specific issues such as degradation of call quality due to low bandwidth, and voice on top of keystroke sounds.

Impact of Typist.

We report here a detailed breakdown of results by different typists on the *Complete Profiling* scenario in Figure 14. Without loss of generality we only consider keystrokes recorded through internal microphones of laptops, to ease presentation of the results. We observed the same tendencies that we report over all the different recording devices.

On these considered datasets, all subjects typed with all their fingers (*Touch typing*), however, as observed in Section 5.2, their typing speed is different. We observe that *S&T attack* often performs better on the slowest typist: accuracy is from 10 % to 35% better on Subject 4, who is the second slowest participant, and around 10% better on Subject 8, who had a non-uniform typing speed. However, Subject 7 who is the slowest is more susceptible only on 2 out of 4 keyboards, with the notable exception of the Fujitsu USB keyboard, where he is the worst performing. Another notable exception is with HP 9009, where the fastest subject is the best performing. Overall, we observe that the impact of typing speed is variable and, in most cases, accuracy is bounded within a few percentage points.

Impact of Recording-device.

We report a detailed breakdown of the impact of different recording-devices, divided by target keyboard, in Figure 15. Performance of *S&T attack* are consistent across different recording-devices. In particular, accuracy on some considered keyboards (HP 9009, HP 250 G5, Dell Inspiron 5000, Fujitsu KB900) is very similar across recording-devices— for example, the HP 9009 only has 3.5% difference in top-5 accuracy on average. Our results do not show significant difference between external and integrated keyboards, showing how *S&T attack* is feasible if the keyboard is not physically part of the same hardware of the microphone. The best performance is usually achieved by using a headset or the laptop itself as recording-device— with the notable exception of the HP KU-0316 USB keyboard, where the smartphone is slightly outperforming the other recording-devices.

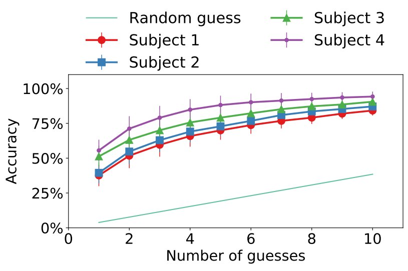
These results further prove that it is not necessary for *S&T attack* to have strong physical proximity between recording-device and target-device (or to sit in the same enclosing, as it is the case for laptops). Another important outcome of these experiments is the extremely good performance of *S&T attack* when using an inexpensive headset microphone as recording-device: indeed, its results are comparable, or even better, than the laptop's integrated microphone, and to the smartphone.

We believe that these results further prove the dangerousness of *S&T attack*. First, one could feel more inclined to type on his keyboard and multi-task during a Skype call if he's using a smartphone or a headset to do it — as the two devices are not the same. However, our results prove that smartphones and headset microphones are as good as the laptop's microphone in leaking keystroke sounds. Second, the effectiveness of eavesdropping through microphones shows that it could be possible to attack a **third party's** keyboard: someone who is only close to the user having the call, who could easily think to be immune to eavesdropping.

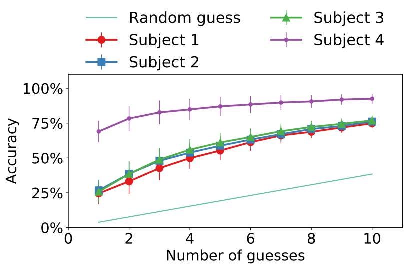
Impact of Small Training Set.

As discussed in Section 3.2, one way to mount *S&T attack* in the *Complete Profiling* scenario is by exploiting data accidentally disclosed by the victim, e.g., via Skype instant-messaging with the attacker during the call. However, our different training datasets have an important shortcoming: they do not respect the letter frequency distribution of any language, as models are trained with a balanced amount of letters — 10 samples for each letter. Ideally, the attacker should be able to mount *S&T* even if the training data is limited to a handful of chat messages. Therefore, to understand the impact of realistic letter frequency distribution, and of the attacker being able to collect just a few sentences together with their ground truth from the victim, we operated as follows:

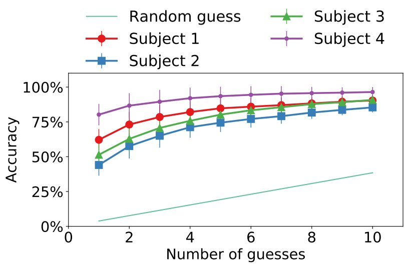
- **"A-Z" dataset:** to account for realistic letter frequency distribution, we retained 10 samples of the most frequent letters according to the Oxford Dictionary [2]. Then, we randomly excluded samples of less frequent letters until only one sample for the least frequent letters



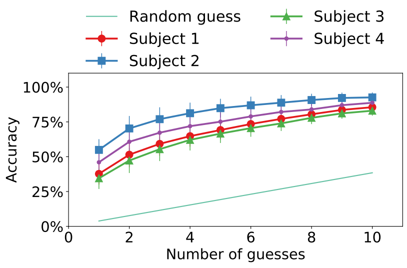
(a) AUKEY USB mechanical keyboard.



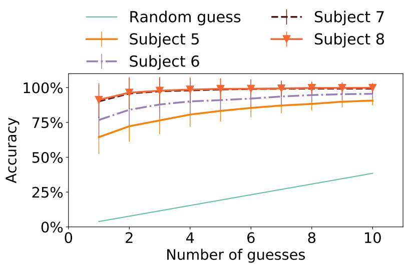
(b) Dell XPS 9333.



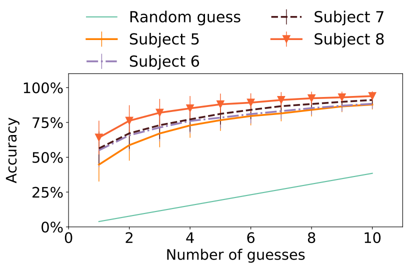
(c) HP HP 250 G5.



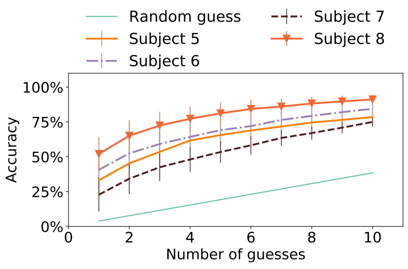
(d) HP 9009 USB keyboard.



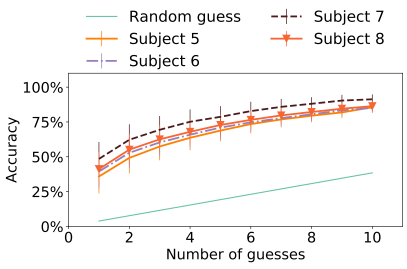
(e) Asus Zenbook UX303UB.



(f) Dell Inspiron 13 5000 series.

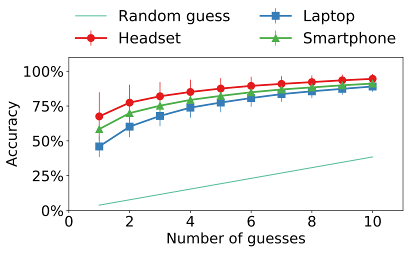


(g) Fujitsu USB keyboard.

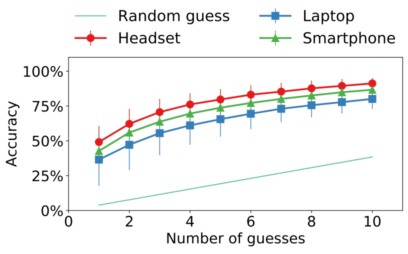


(h) HP USB keyboard.

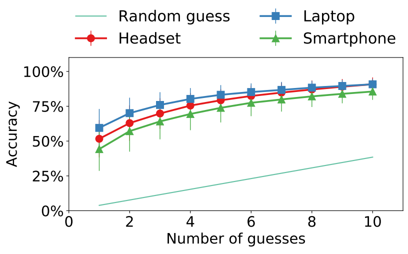
Fig. 14. *S&T attack* performance — *Complete Profiling* scenario, Touch typing, Skype-filtered data; average accuracy for different typists.



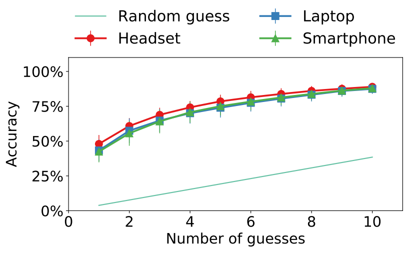
(a) AUKEY USB mechanical keyboard.



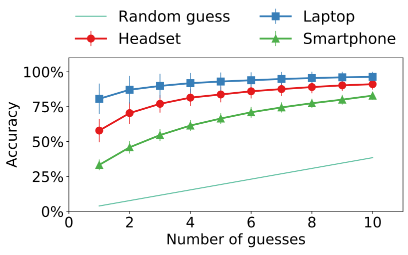
(b) Dell XPS 9333.



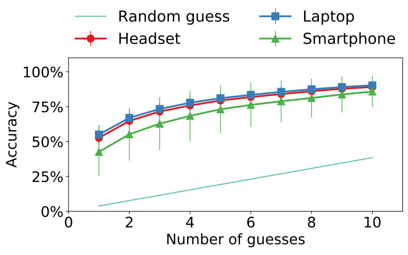
(c) HP 250 G5.



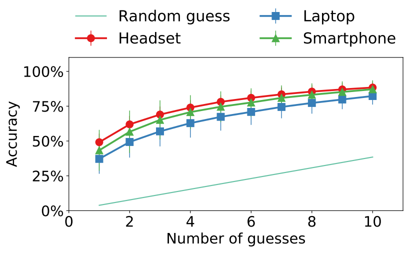
(d) HP 9009 USB keyboard.



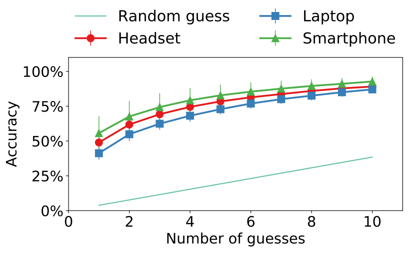
(e) Asus Zenbook UX303UB.



(f) Dell Inspiron 13 5000 series.



(g) Fujitsu KB900 USB keyboard.



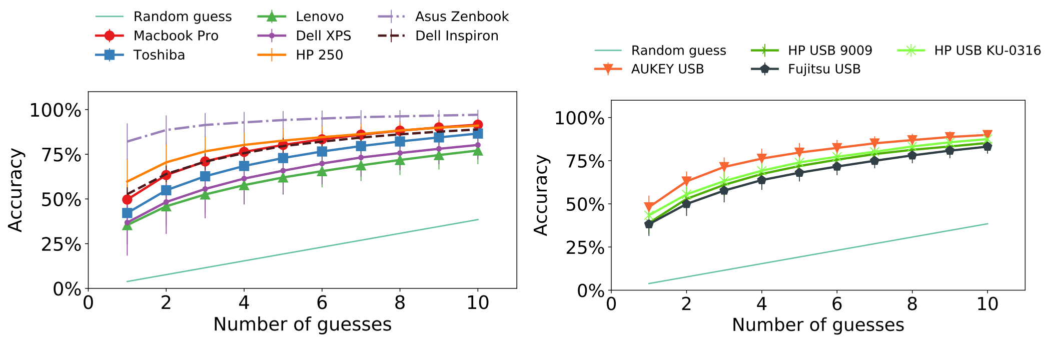
(h) HP KU-0316 USB keyboard.

Fig. 15. *S&T attack* performance — *Complete Profiling* scenario, Touch typing, Skype-filtered data; average accuracy for different recording devices.

was available. Ultimately, the subset contained 105 samples, that might correspond to a typical short chat message or a brief email. We then evaluated performance of the classifier trained with this subset, on a 10-fold cross-validation scheme. This random exclusion scheme was repeated 20 times for every fold.

- **"Quick Brown Fox" dataset:** As this dataset is composed by English sentences, we don't need to alter its letter frequency distribution. However, to reduce its size, we randomly selected 4 sentences out of the 10 available for each subject and device: 3 sentences (135 characters) form the training set — one, the test set. We then evaluated performance of the classifier. This random exclusion scheme was repeated 20 times for every laptop and user combination.

We show results on Touch typing, Skype filtered data in Figure 16.



(a) Laptop Keyboards. (b) USB Keyboards.

Fig. 16. *S&T attack* performance — average accuracy when training on a small subset of samples that respects the letter frequency of the English language.

We observe that, with both datasets, there is an acceptable top-5 accuracy loss of 10%-25%, depending on the specific model and user. This mainly happens because now the training sets are very small (105 and 135 characters, respectively), and the less frequent letters for which we have only a few samples in the training set are harder to classify. However, with an impressively small training set, *S&T attack* still provides very good performance. There is an evident outlier: the Asus Zenbook laptop performs almost as good as with the full training set, despite the reduced size of the training set. These results further motivate the *Complete Profiling* scenario: the attacker can exploit even a few acoustic emanations that the victim discloses via a short message during a Skype call.

Impact of Fluctuating VoIP Bandwidth.

A prominent issue that stems from using VoIP to perform *S&T attack*, is that the SILK codec [38] degrades performance of *S&T attack*. For example, this codec reduces audible bandwidth whenever available Internet bandwidth is low, in order to use less bandwidth. This useful technical feature can, however, harm accuracy of *S&T attack*, because it degrades the sound spectrum.

In the experimental setup, both VoIP end-points were connected to a high-speed network. However, a realistic call might go over slower or more error-prone network links. Therefore, we performed a number of sample Skype calls between the two end-points while monitoring network load of the transmitter (i.e., the one producing emanations).

We experimented as follows: we filtered all data recorded on one Macbook Pro laptop by all the users with the HP typing style using Skype, together with a five minutes sample of the *Harvard Sentences*, commonly used to evaluate the quality of VoIP applications [35]. We initially let the Skype software use the full bandwidth available, and we measured that the software used an average of 70 Kbit/s without any noticeable packet loss. We subsequently limited the bandwidth of the transmitting machine at 60 Kbit/s, 50 Kbit/s, 40 Kbit/s, 30 Kbit/s, respectively, 20 Kbit/s. We observed that, with values below 20 Kbit/s, the quality of the call is compromised, because of frequent disconnections. *S&T attack* with such a small bandwidth is therefore not possible, and we argue that real users suffering this degradation of service would anyway not be willing neither able to continue the Skype call. Therefore, we believe the bandwidths we selected are representative of all the conditions on which we find the Skype software is able to operate. We then evaluated both the accuracy of *S&T attack*, and the quality of the call by using the voice recognition software CMU Sphinx v5 [28] on the Harvard Sentences. We show the results in Figure 17.

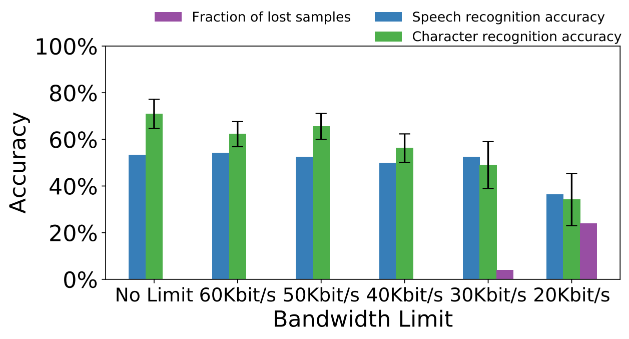


Fig. 17. Voice recognition and *S&T attack* accuracy, on data acquired through Skype with different connection bandwidths.

From Figure 17, we can see that, while there is no change to the accuracy of the voice recognition software until the 20 Kbit/s threshold, the classifier suffers a noticeable loss at and under 40 Kbit/s. This analysis shows that aggressive downsampling, and communication errors, can greatly hinder the accuracy of the attacker on the eavesdropping task, and that a loss of the order of 20% is to be expected if the connection speed is very low. We also observe that, at 20 Kbit/s, even if the Skype call is working, many samples of both the speech and keyboard sounds are lost or irreparably damaged due to the small bandwidth, and the final quality of the call might be undesirable for the user. However, it is realistic to assume Skype to be always working at the best possible quality or almost at the best possible quality, since 70-50 Kbit/s are bandwidths that are small enough to be almost guaranteed.

Impact Of Voice.

In the experiments we described so far, we did not consider that the victim can possibly be talking while he types target-text. However, in a VoIP call, this can happen frequently, as it is probable that the victim is talking while he types something on the keyboard of his target-device. We evaluated the impact of this scenario as follows: we considered all the data of one user on the Macbook Pro laptop, consisting of 260 samples, 10 for every class, in a 10-fold cross-validation scheme. As usual, we used 9 folds to train a model as described in Step C (see Section 4). We then overlapped the sounds of the last fold (i.e., the test data) with random portions of a recording of a male voice spelling some Harvard Sentences [35]. The overlap was done by summing the amplitude of the waveforms, and taking care that it was not exceeding the maximum intensity allowed by its discrete representation. We underline that the algebraic sum of the amplitudes of two waveforms leads to a very good approximation of recording the two sounds on the same microphone at the same time. This physical phenomenon is known as superposition of waves (also defined as wave interference), that happens when two waves are incident on the same point (in our case, the microphone) — we refer the interested reader to the definition of acoustic interference in [11]. To account for the random overlap, we repeated the process 10 times, to have the keystroke sound overlap different random phonemes. We then evaluated the mean and standard deviation of the accuracy of the classifier.

We repeated the described experiment with different relative intensities of the voice against the intensity of the sound of the keystrokes. We started at -20dB, meaning that the keystrokes are 20dB louder than the voice of the speaker, and evaluated progressive steps of 5dB, until we had the voice of the speaker 20dB louder than the keystrokes. We performed this scheme on the data for all users on the Macbook Pro laptop, with Touch typing and data filtered with Skype. We show the results in Figure 18.

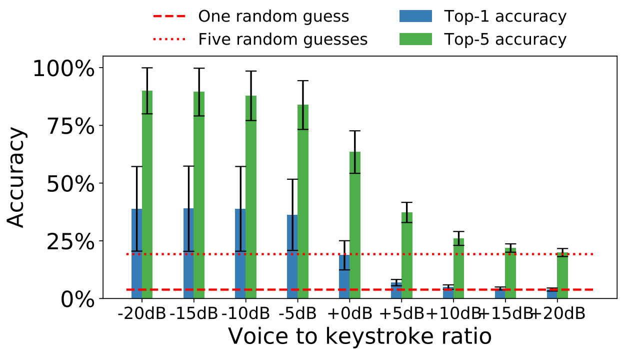


Fig. 18. *S&T attack* performance — average accuracy, overlap of keystroke sounds and voice, at different relative intensity.

We observe that, from −20dB until 0dB, *S&T attack* does not suffer almost any performance loss, and then the accuracy rapidly decreases, until it reaches the random guess baseline at +20dB. We explain both the positive and the negative results with the phenomenon of auditory masking [41], where only the most powerful tone among all the tones at a given frequency is audible. This phenomenon is a direct consequence of wave interference. In our case, the greater the difference between the intensity of the sound of the keystroke and of the voice, the more only the frequencies of the louder sound will be audible.

We believe that the takeaway of our experiment is the following: keystrokes that are still audible can be predicted by S&T, while keystrokes that are covered by voice cannot, as too much information is lost because of wave interference. Given that the keystrokes are very loud when recorded from a laptop microphone, or in close proximity of the keyboard, it is likely that most keystrokes can be recovered despite the presence of voice.

6.5 Discussion and Limitations

To corroborate our claims, we evaluate the statistical significance of our obtained results. To do so, we applied a statistical test that measures how likely the results could have been generated by chance: the *binomial test* [16]. This statistical test allows to model scenarios where the observations are divided into two categories. In our case, we consider the two categories to be the results of *S&T* on our baseline scenario (Complete Profiling), and the baseline random guess. The null hypothesis is that our results are generated by chance — and thus the two categories are similarly likely. We first modeled the baseline random guess observation as a binomial distribution $B(n, p)$ with the following parameters: n (the number of trials) corresponds to the number of collected sample for a single letter — 320; while p (the probability of success on a single trial) corresponds to the top-10 accuracy of the baseline — 10/26.

We then perform the binomial test to compare our observation (the accuracy of *S&T*) to this binomial distribution modeling the baseline: we evaluate the probability $P(X > x)$, where X is the distribution B(n,p) and x is the minimum achieved score obtained by our attack. The minimum achieved score in our attack is 0.5. We then calculate $P(X > 0.5) = 1.09709 \cdot 10^{-5}$, allowing us to refuse the null hypothesis for $p < 0.001$ — thus showing statistical significance of our results.

The calculation of the binomial test, combined with all our results on different settings, target-devices, and recording-devices, shows that *S&T attack* is a provable threat for a large population of keyboard users. However, as previously discussed in Section 5.2, our study has some limitations. While we believe that good typists are somehow the *worst* case for *S&T attack*, as observed by highest accuracy obtained by the slowest typists, our experimental population is composed of mostly young people that have a certain degree of confidence with keyboards and computers. There are hints that *S&T* is applicable to less expert typists, as our HP typing style experiments tried to simulate the typing patterns of less accustomed users; however, we cannot affirm with certainty if our results generalize to greatly different populations, e.g., older users with very low proficiency with a keyboard.

7 S&T PRACTICAL APPLICATIONS

We now consider two practical applications of the results of *S&T attack*: understanding words, and cracking random passwords. In particular, if the victim is typing English words, we analyze how *S&T* can help understanding such words. If the victim is typing a random password, we show how *S&T attack* can greatly reduce the average number of trials required in order to crack it, via a brute force attack. In the following, we report the results of these practical applications on the *Complete Profiling* scenario, and on the *Model Profiling* scenario.

7.1 Word Recognition

To evaluate how *S&T* helps understanding the words that the victim typed, we proceeded as follows. We drew a number of random words from an English dictionary; we call such words *actual words*. For each *actual word*, we reconstructed its typing sound combining the sound samples of each letter in the actual word. We used the sound sample of the letters we collected in Section 5.1. We then performed *S&T attack*, to obtain the top-10 predictions for each letter of the actual word. We combined these predictions with an English dictionary, by sorting it according to each word's probability given the predictions from *S&T*, and retain the first 20 words: the most probable *guessed words* for the actual word. We then count how many times the *actual word* is contained in the 20 most probable guessed words. Furthermore, we calculated the number of correctly guessed characters as the minimum of the Hamming distances between the actual word and the guessed words.

We report in the following the results of our experiments on 1000 random English words, on the *Complete Profiling* scenario. We show in Figure 19 how many times the actual word was in the set of guessed words. In particular, for 4-characters long words, we were able to recover 71% of them on the Lenovo laptop, up to 98% of the words on the MacBook Pro laptop. The accuracy increases for longer words, because the impact of wrong guesses is mitigated by the presence of more characters. From 10-characters long words, on MacBook Pro and Toshiba, we were able to recover 100% of the words. For Lenovo, from 15-character words.

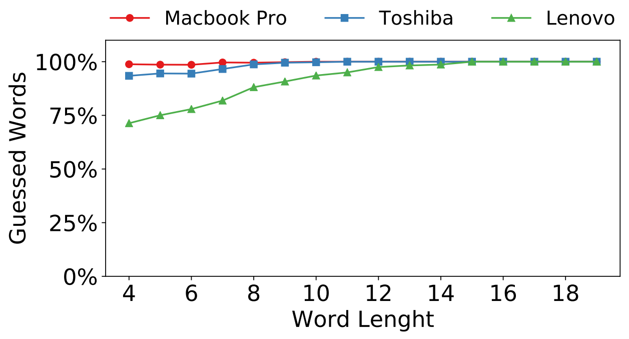


Fig. 19. *S&T attack* performance — fraction of entirely correct recovered words of different lengths, by combining *S&T attack* guesses with a dictionary.

Figure 20 shows the average amount of correctly guessed characters. We observe that on the Toshiba laptops the average number of correctly recovered letters of the word is around 50% for words of length 4-7, and then rapidly increases. Average correctness of the recovered word for Lenovo is between 80% and 90%, and rapidly approaches 100% from words of length 10, as discussed in Figure 19. With Macbook Pro laptops, *S&T* almost always recovers all the letters correctly, consistently with its very high results in the *Complete Profiling* scenario.

We believe these results highlight the threat posed by *S&T attack*. Indeed, when recovering text in a known language (e.g., English), additional information on words amplifies the results of *S&T*, allowing almost perfect word recovery in many settings. Furthermore, as shown by the confusion matrices in Figure 9, misclassifications often happen among neighboring keys — a language model could further be expanded to weight more predictions corresponding to neighbors of the first few guesses.

7.2 Password Recognition

Secure passwords that prevent dictionary attacks are random combinations of alphanumeric characters. In this section, we shed some light on how *S&T* can help cracking random password with an improved brute-force scheme that takes advantage of our results. We distinguish between two cases: characters that are typed with a single keystroke (i.e., lowercase letters, numbers, some symbols, the caps-lock key, and uppercase letters during caps-lock), and characters that are typed with two keystrokes, Shift + a key (i.e., symbols, and uppercase letters with Shift). We first detail

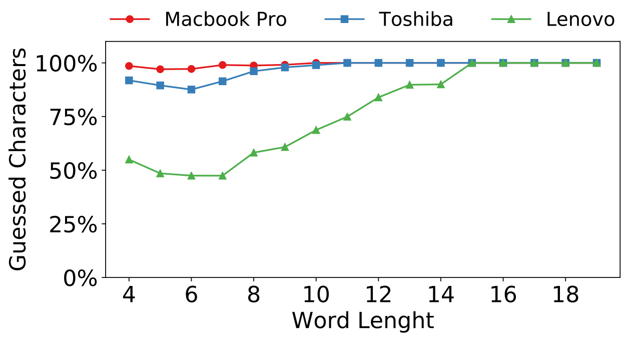


Fig. 20. *S&T attack* performance — fraction of correctly guessed characters for not completely correct recovered words, by combining *S&T attack* guesses with a dictionary.

a scheme that works ok single-keystroke characters, as it reflects our experimental results, and calculate the speedup of the improved brute-force scheme for such keys. We then outline how a more complete scheme could consider the Shift key.

The improved brute-force scheme for single-keystroke characters is as follows: given the x guesses of *S&T* for each of the n characters of the target password, we first consider all the x^n combinations of such characters. We then assume that the set of x guesses of the first character was wrong, and subsequently consider all the other characters. When we finish considering that one set of guesses was wrong, we consider all the combinations of two wrong guesses (i.e., first and second sets of guesses were wrong, first and third sets were wrong, up to the seventh and eighth sets). We repeat this scheme until we finally try the combinations where the classifier was always wrong. This brute-force scheme leverages the probability of success of *S&T* to minimize, on average, the required time to crack a password. If we consider a target password of 10 lowercase characters of the English alphabet, a regular brute-force scheme requires requires $\frac{(26)^{10}}{2} = 8.39 \cdot 10^{13}$ guesses to have 50% probability. On the *Complete Profiling* scenario, that we recall has an average top-5 accuracy of more than 90%, we only need $9.76 \cdot 10^6$ tries to have 50% probability. This corresponds to a very high average speedup of 10^7, and an entropy reduction of more than 50%. On the *Model Profiling* scenario, where we have a top-5 accuracy around 40%, we need $7.79 \cdot 10^{12}$ tries to reach 50% probability of cracking the password, which is still one order of magnitude better than plain brute-force attacks, on average. There is similar tendency if the attack guesses ten characters for every character of the password. These results are calculated on letters only (our collected dataset) — but other single-keystroke keys are similarly detectable by S&T, given training data.

To consider characters that are typed with two keystrokes, an improved scheme would need to precisely pinpoint the pressure and release of the Shift key. We leave the development of a complete acoustic-enhanced password cracker as future work, and here hint on the following fact. As discussed in Section 4.2, keystroke sounds have a *press* and a *release* peak. Users keep the Shift key pressed down to type special characters — therefore, the attacker could detect the Shift key by finding keystrokes that have press and release peak separated by other keystrokes. This consideration can be used to deduct that properly-detected subsequent single key presses are instead the alternate character of such key. Another possibility is to train a dedicated attacker model to detect the Shift key only. We leave this as future work.

8 COUNTERMEASURES

In this section, we present and discuss some potential countermeasures and analyze their efficacy in preventing *S&T* and other attacks that use statistical properties of the sound spectrum. While we discussed some countermeasures in our preliminary version of this work [17], here we provide a complete countermeasure, able to completely prevent attacks based on spectral properties of keystroke sounds.

First, we argue that designing good countermeasures to keyboard acoustic eavesdropping attacks, such as *S&T*, is difficult. This is because the underlying goal for the user is to perform high-quality, undisturbed VoIP calls. However, many countermeasures impact VoIP call quality. For example, one simple countermeasure is applying a short "ducking" effect, a technique that drastically lowers microphone volume and overlaps it with a different sound, whenever a keystroke is detected. However, this approach would greatly degrade voice call quality. Another possible countermeasure, proposed in [7], is to introduce fake keystroke sounds during the call, when the user is typing. However, we note that this countermeasure requires training with keystroke sounds of the specific laptop in use, and users might find it annoying, as the amount of noise increases greatly. We believe that ideally, an effective countermeasure should be minimally intrusive and affect only keystroke sounds.

Moving from these observations, we design a less intrusive countermeasure against all techniques that use sound spectrum information — performing short random transformations to the sound whenever a keystroke is detected. One intuitive way to do this is to apply a random multi-band equalizer over a number of small frequency bands of the spectrum. This allows us to modify the intensity of specific frequency ranges, called "bands". Each band should be selected at random and its intensity should be modified by a small random amount, thus effectively changing the sound spectrum. All these random decisions should change every few seconds, to prevent the adversary from learning the modifications. An alternative approach, to reduce intrusiveness, can be to trigger the countermeasure only when the OS detects a keystroke.

To show the efficacy of this countermeasure, we ran the following experiment: we considered, as usual, our datasets in a 10-fold cross-validation scheme. As detailed in Section 4, we trained the classifier (Step C) with 9 folds. However, before evaluating the accuracy of the attacker, we transformed the test data as follows: we applied a multiband equalizer with 50 bands to the test data only, where each band has a random center between 100 Hz and 12,000 Hz, a very high resonance factor Q of 20 (meaning that the gain of each band has impact only on neighboring frequencies), and a random gain between -5dB and +5dB. We then measured the accuracy of the classifiers on the test data processed by our countermeasure.

We report that our countermeasure proves extremely effective: top-1 accuracy of the test data filtered by the countermeasure is 3.75%–6.18% (random guess baseline is 3.84%) and top-5 accuracy is 18.70%–24.70% (baseline is 19.23%). Even top-10 accuracy is greatly impacted: 37.38%–44.94% (baseline is 38.46%). This shows that *S&T* has no sensible advantage over random guesses after applying our countermeasure to keystroke sounds.

Our approach also allows the speaker's voice to remain intelligible. To show this, we operated as follows: we selected a corpus of recordings of 30 spoken sentences, and used the IBM Watson speech-to-text engine to extract the content of the sentences. We then applied our countermeasure on the sentences, and used again IBM Watson to obtain their transcripts. To account for the randomness in the countermeasure, we repeat the process 5 times and average the results. We report a very high similarity between the transcripts before and after the countermeasure: 93.69%(±11.89%). This is further confirmed by considering the average Pearson correlation between recordings before and after our countermeasure: 86.35%. As a final validation, we listened to the resulting files and confirm the acceptable quality — we uploaded an archive with the original files and their transformations by the countermeasure on the project webpage[6].

As a final experiment, we tested whether this countermeasure is effective against different spectral features: we repeated our experiment extracting both MFCC and FFT features, applying the countermeasure on test data. Results in Figure 21 show *S&T* accuracy, with and without the countermeasure, for MFCC and FFT features.

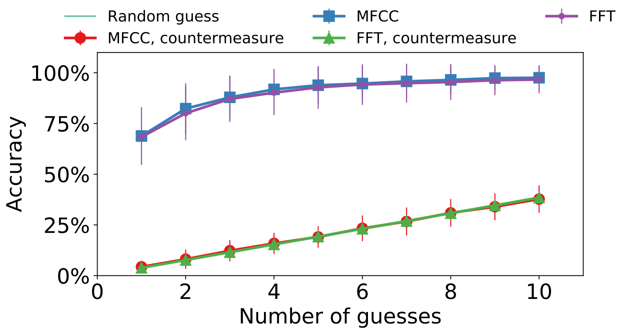


Fig. 21. Average accuracy of single key classification against a random equalization countermeasure. Accuracy of both FFT and MFCC features after countermeasure lies exactly on the random guess baseline.

We observe that our proposed countermeasure also successfully disrupts FFT features (used, e.g., in [8, 23, 24, 33], besides MFCC features (used in this work, and in [7, 45]). Indeed, the accuracy of *S&T* goes down to the baseline random guess for both sound features.

A more simplistic approach is to use software or emulated keyboards, i.e., those that appear on the screen and are operated by the mouse. Similarly trivial ideas include: (1) activating a mute button before typing, or (2) not to type at all whenever engaged in a VoIP call.

9 CONCLUSIONS

We demonstrated a highly accurate VoIP-based remote keyboard acoustic eavesdropping attack. We first described a number of practical attack scenarios, using VoIP as a novel means to acquire acoustic information under realistic assumptions: random target text and very small training sets, in Section 3. Then, in Section 4 we demonstrated an attack with these assumptions in mind and carefully selected the tools to maximize its accuracy. In Section 5 we detail the data we collected to then thoroughly evaluate *S&T attack* in several scenarios in Section 6. In Section 7 we showed that *S&T attack* can be used to recover both random and non-random text, and finally we discussed some potential countermeasures to *S&T* and other attacks that leverage spectral features of keyboard sounds, in Section 8.

We believe that this work, due to its real-world applicability, advances the state-of-the-art in acoustic eavesdropping attacks. *S&T attack* was shown to be both feasible and accurate over Skype, in all considered attack scenarios, with none or minimal profiling of the victim's typing style and keyboard. In particular, it is accurate in the *Model Profiling* scenario, where the attacker profiles a laptop of the same model as the victim's laptop, without any additional information about the victim. This allows the attacker to learn private information, such as sensitive text or passwords. We also took into account VoIP-specific issues – such as the impact of audible bandwidth reduction, and effects of human voice mixed with keystroke audio – and showed that *S&T* is robust with

[6]https://spritz.math.unipd.it/projects/dst

respect to both. Finally, we discussed some countermeasures and concluded that *S&T* is hard to mitigate.

Future Work. The high accuracy of *S&T* highlights the importance of robust countermeasures to such sound-based side channels. Our proposed countermeasure is able to completely stop *S&T* (and other attacks that leverage spectral properties of keystroke sounds). However, as it might impact the call quality, more thorough user studies are needed to validate our preliminary claim that voice is not affected by the countermeasure. Another interesting direction might be, instead of random equalization in hopes of disrupting important features, leveraging results on adversarial machine learning and transferability to generate the minimal perturbations that cause attacks such as *S&T* to misclassify keystroke sounds.

ACKNOWLEDGMENTS

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the LOCARD project (Grant Agreement no. 832735).

REFERENCES

[1] Opus codec support. URL https://wiki.xiph.org/OpusSupport.

[2] Oxford dictionary - which letters in the alphabet are used most often. URL http://www.oxforddictionaries.com/words/which-letters-are-used-most.

[3] 2015: Skype's year in review, . URL http://blogs.skype.com/2015/12/17/2015-skypes-year-in-review/.

[4] Over 1 billion skype mobile downloads, . URL http://blogs.skype.com/2016/04/28/over-1-billion-skype-mobile-downloads-thank-you/.

[5] Microsoft build 2016 keynote, . URL https://channel9.msdn.com/Events/Build/2016/KEY01.

[6] K. Ali, A. Liu, W. Wang, and M. Shahzad. Keystroke recognition using wifi signals. In *ACM MobiCom*, pages 90–102, 2015.

[7] S. A. Anand and N. Saxena. Keyboard emanations in remote voice calls: Password leakage and noise (less) masking defenses. In *ACM CODASPY*, 2018.

[8] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE S&P*, pages 3–11, 2004.

[9] K. S. Balagani, M. Conti, P. Gasti, M. Georgiev, T. Gurtler, D. Lain, C. Miller, K. Molas, N. Samarin, E. Saraci, et al. Silk-tv: Secret information leakage from keystroke timing videos. In *European Symposium on Research in Computer Security*, pages 263–280. Springer, 2018.

[10] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from video. In *IEEE S&P*, pages 170–183, 2008.

[11] D. Basu. *Dictionary of pure and applied physics*. CRC press, 2000.

[12] Y. Berger, A. Wool, and A. Yeredor. Dictionary attacks using keyboard acoustic emanations. In *ACM CCS*, pages 245–254, 2006.

[13] S. Boyd, C. Cortes, M. Mohri, and A. Radovanovic. Accuracy at the top. In *NIPS*, pages 953–961, 2012.

[14] S. Card, T. Moran, and A. Newell. The keystroke-level model for user performance time with interactive systems. *CACM*, 23(7):396–410, 1980.

[15] Y. Chen, T. Li, R. Zhang, Y. Zhang, and T. Hedgpeth. Eyetell: video-assisted touchscreen keystroke inference from eye movements. In *IEEE S&P*, pages 144–160, 2018.

[16] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[17] A. Compagno, M. Conti, D. Lain, and G. Tsudik. Don't skype & type!: Acoustic eavesdropping in voice-over-ip. In *ACM ASIACCS*, 2017.

[18] A. Das, N. Borisov, and M. Caesar. Do you hear what I hear?: fingerprinting smart devices through embedded acoustic components. In *ACM CCS*, pages 441–452, 2014.

[19] S. Fang, I. Markwood, Y. Liu, S. Zhao, Z. Lu, and H. Zhu. No training hurdles: Fast training-agnostic attacks to infer your typing. In *ACM CCS*, pages 1747–1760, 2018.

[20] J. Friedman. Tempest: A signal problem. *NSA Cryptologic Spectrum*, 1972.

[21] D. Genkin, M. Pattani, R. Schuster, and E. Tromer. Synesthesia: Detecting screen content via remote acoustic side channels. *arXiv preprint arXiv:1809.02629*, 2018.

[22] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

[23] T. Halevi and N. Saxena. A closer look at keyboard acoustic emanations: random passwords, typing styles and decoding techniques. In *ACM CCS*, pages 89–90, 2012.
[24] T. Halevi and N. Saxena. Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios. *International Journal of Information Security*, 14(5):443–456, 2015.
[25] K. Jin, S. Fang, C. Peng, Z. Teng, X. Mao, L. Zhang, and X. Li. Vivisnoop: Someone is snooping your typing without seeing it! In *IEEE CNS*, pages 1–9, 2017.
[26] T. Kaczmarek, E. Ozturk, and G. Tsudik. Thermanator: Thermal residue-based post factum attacks on keyboard password entry. *arXiv preprint arXiv:1806.10189*, 2018.
[27] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. *IEEE TDSC*, 2(2):93–108, 2005.
[28] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf. The cmu sphinx-4 speech recognition system. In *IEEE ICASSP*, volume 1, pages 2–5, 2003.
[29] J. Liu, Y. Wang, G. Kar, Y. Chen, J. Yang, and M. Gruteser. Snooping keystrokes with mm-level audio ranging on a single phone. In *ACM MobiCom*, pages 142–154, 2015.
[30] B. Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, 2000.
[31] J. Lukas, J. Fridrich, and M. Goljan. Digital camera identification from sensor pattern noise. *IEEE TIFS*, 1(2):205–214, 2006.
[32] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp) iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *ACM CCS*, pages 551–562, 2011.
[33] Z. Martinasek, V. Clupek, and K. Trasy. Acoustic attack on keyboard using spectrogram and neural network. In *TSP*, pages 637–641, 2015.
[34] J. Monaco. Sok: Keylogging side channels. In *IEEE S&P*, 2018.
[35] E. Rothauser, W. Chapman, N. Guttman, K. Nordby, H. Silbiger, G. Urbanek, and M. Weinstock. Ieee recommended practice for speech quality measurements. *IEEE Transactions on Audio and Electroacoustics*, 17(3):225–246, 1969.
[36] D. Shukla, R. Kumar, A. Serwadda, and V. Phoha. Beware, your hands reveal your secrets! In *ACM CCS*, pages 904–917, 2014.
[37] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001.
[38] J.-M. Valin, K. Vos, and T. Terriberry. Definition of the opus audio codec. *IETF, September*, 2012.
[39] M. Vuagnoux and S. Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *USENIX Security*, pages 1–16, 2009.
[40] 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*, pages 14–27, 2014.
[41] R. Wegel and C. Lane. The auditory masking of one pure tone by another and its probable relation to the dynamics of the inner ear. *Physical Review*, 23(2):266, 1924.
[42] T. Wei, S. Wang, A. Zhou, and X. Zhang. Acoustic eavesdropping through wireless vibrometry. In *ACM MobiCom*, pages 130–141, 2015.
[43] W. Wodo and L. Hanzlik. Thermal imaging attacks on keypad security systems. In *ICETE*, 2016.
[44] T. Zhu, Q. Ma, S. Zhang, and Y. Liu. Context-free attacks using keyboard acoustic emanations. In *ACM CCS*, pages 453–464, 2014.
[45] L. Zhuang, F. Zhou, and D. Tygar. Keyboard acoustic emanations revisited. *ACM TISSEC*, 13(1):3, 2009.