



OverHear: Headphone Based Multi-Sensor Keystroke Inference

Raveen Wijewickrama¹(✉) , Maryam Abbasihafshejani¹ , Anindya Maiti² ,
and Murtuza Jadliwala¹

¹ The University of Texas at San Antonio, San Antonio, TX, USA

{raveen.wijewickrama,maryam.abbasihafshejani,murtuza.jadliwala}@utsa.edu

² The University of Oklahoma, Norman, OK, USA
am@ou.edu

Abstract. Headphones, once used exclusively for audio playback, now feature advanced sensors such as high-definition microphones and accelerometers. The integration of these sensors has enabled new functionalities; however, they have also introduced new security/privacy risks. This work explores the feasibility of inferring user keystrokes on external keyboards through sensory side-channels present on modern headphones. Specifically, we develop a novel keystroke inference framework, called *OverHear*, which uses both acoustic and motion data available from modern headphones to infer user keystrokes on external keyboards. In *OverHear*, the motion or accelerometer data is used to cluster key presses by hand position, while the acoustic data is employed to distinguish between different keystrokes through an analysis of the extracted Mel Frequency Cepstral Coefficients (MFCC). Trained machine learning (ML) models and a dictionary-based word prediction approach are then used to identify individual keystrokes and typed words. Comprehensive experiments on a variety of commercial keyboards in realistic settings show that *OverHear* achieves close to 80% top-5 key prediction accuracy for mechanical keyboards, 60% for membrane keyboards, and more than 70% top-100 word prediction accuracy for all types of keyboards.

Keywords: headphones · wearable · privacy · keystroke inference

1 Introduction

Headphones (or earphones) have become ubiquitous computer peripherals that offer private and immersive experiences across various online communication and entertainment applications. Modern headphones are equipped with an array of sensors, including *accelerometers* for motion detection and high definition *microphones* for high-fidelity voice capture. These sensors enable advanced functionalities on headphones such as wearing detection [4], gesture controls [12, 14], adaptive noise cancellation [38], spatial audio [25], and fitness tracking [29, 31]. Nevertheless, due to their constant placement near the user's body, these contemporary headphones raise significant security and privacy issues. Their location

on the user’s body and their advanced sensing features might offer a unique vantage point that could be exploited to discreetly gather sensitive information related to the user and his/her activities. In this work, we are interested in investigating the feasibility of spying on users’ keystrokes (on external keyboards and keypads) by using the information available from headphones (specifically, headphone sensors) as an information side channel.

Although prior work on keystroke inference through acoustic eavesdropping has leveraged devices such as smartphones placed near keyboards [8, 21] or relied on motion sensors in wearables (e.g., smartwatches) [24], these approaches typically require deliberate device placement or close physical proximity to the victim’s keyboard. Diverging from these traditional keystroke inference attack scenarios, which typically involve unpredictable and non-practical placement of explicit recording devices near the target user, we investigate the feasibility of a more realistic scenario involving a recording device (i.e., a headphone) that is ubiquitously present on the target user in a predictable location. In particular, many Bluetooth-enabled headphone manufacturers [3, 18, 33] provide companion applications that can pair with headphones and potentially gain access to raw sensor data. This access could be exploited by malicious adversaries who seek to stealthily record user interactions and infer keystrokes.

Nevertheless, there are several other non-trivial challenges (in such a keystroke inference attack scenario) which will need to be overcome. One challenge is that the relative position between the headphones and the keyboard can vary considerably based on user traits such as height, arm length, and distance from the chair to the table. This is further compounded by voluntary and involuntary user head movements during typing, which can be perceived by headphone sensors. Prior studies [23, 45] on keystroke inference using sensory side channels frequently utilized a pair of microphones to calculate the Time Difference of Arrival (TDoA) of sound signals. These techniques have produced favorable results, largely due to the fact that the data collection device was located on the same surface as the keyboard or served as the input device itself (for example, keystroke inference on a mobile phone keypad) [44]. In addition, these previous works on acoustic keystroke inference mainly relied on recording devices that were *stationary*. As users’ head movements induce motion of the headphone, the recording device (i.e., headphone) can no longer be considered to be stationary and may record much more noisier/inconsistent data.

In light of the above challenges, in this paper we propose *OverHear*, a novel framework to accurately infer keystrokes using acoustic and motion sensor data collected from headphones. We observe that while the headphone motion data (as sensed by the device accelerometer) alone is not enough to distinguish individual keystrokes, it proves to be very useful in clustering keys corresponding to each hand. To predict individual keystrokes, we take advantage of the headphone microphone data. We first extract Mel-Frequency Cepstral Coefficients (MFCC) features from the acoustic data and then use these features to train machine learning-based prediction models. The prediction of these models is then further refined using a dictionary-based spelling correction approach to improve the over-

all inference accuracy of the typed words. We evaluated *OverHear* with the help of 17 participants using a custom headphone hardware setup and 5 participants using commercially available Apple AirPods. Our comprehensive experimental evaluation shows that *OverHear* is able to achieve a top-5 key prediction accuracy (i.e., the correct key is among the top five predicted keys) of approximately 80% for mechanical keyboards and approximately 60% for membrane keyboards. Furthermore, our framework demonstrates a top-100 word prediction accuracy of more than 70% across all keyboard types.

Our main technical contributions are as follows:

- **A novel keystroke inference framework** which employs data from headphone microphones and accelerometers to accurately infer user keystrokes. The framework also includes a keyboard type identification module to identify the keyboard type used by the victim (e.g., mechanical or membrane).
- **An enhanced word prediction mechanism** based on spell-correction to improve overall keystroke inference by mitigating errors caused by missing or extraneous keystrokes.
- **Comprehensive empirical evaluation** using data from human-subject participants in realistic and unconstrained settings, spanning various environmental/ambient noise scenarios.

2 Related Work

Analysis of acoustic signals for the purpose of inferring user input on a variety of mobile and computing devices has been the subject of several studies in recent years, each employing unique methodologies and achieving varying degrees of accuracy. Table 1 provides a summary of these existing works.

Among the foundational studies, the contributions of Asonov and Agarwal [7] and Liu et al. [21], although significant, were tested under rather restrictive conditions and focused solely on individual key presses rather than more complex scenarios involving word or sentence typing and prediction. Zhu et al. [45] also proposed a TDoA-based keystroke inference framework and achieved a key prediction accuracy of 72%. However, their framework requires at least two malicious mobile phones to be in physical proximity to the victim keyboard, which makes executing the attack practically challenging. Ceconello et al. [11] proposed an attack that uses audio data collected during a VoIP application call to infer keystrokes, which is perhaps the closest work to ours in terms of attack surface (hardware) setup. They use microphone data from a laptop, a smartphone and a traditional headset (with a boom microphone; a single microphone protruding near towards the mouth) to evaluate their attack. They highlight an attack scenario where the attacker obtains the victim’s labeled training data through social engineering to achieve a top-5 accuracy of 80%. However, this accuracy drops below 55% when no training data from the victim is available, indicating a potential limitation of their framework due to the reliance of victim’s labeled training data.

Table 1. Summary of related work on acoustic based keystroke inference attacks.

Study	Methodology	Recording Device	Accuracy
Asonov and Agarwal [7]	Frequency domain features + Neural Network	PC microphone	79% (key)
Liu et al. [21]	TDoA	Smartphone (stereo microphones)	85% (key)
Narain et al. [27]	Raw Audio, spatial positioning + Decision Tree	Smartphone	95% (key)
Lu et al. [23]	TDoA	Smartphone	50% (top-1), 90% (top-10) word
Zhu et al. [45]	TDoA	Two smartphones	72% (keys)
Bai et al. [8]	TDoA, PSD + SVM	Smartphone	92% (top-5 key)
Cecconello et al. [11]	MFCC + Logistic Regression	Laptop, smartphone, headset	80% (top-5 key)
Harrison et al. [19]	Mel-spectrograms + DNN	Smartphone	95% (key)

Several past studies have also explored the use of motion sensors for inferring keystrokes [10, 22, 24, 40]. These investigations typically harness motion sensors in mobile phones to deduce in-device keystrokes or employ smart wearables, such as smartwatches, to infer keystrokes on physical keyboards. Specifically, Cai and Chen [10] highlighted the feasibility of motion-based keystroke inference attacks, demonstrating that motion data could be exploited to deduce user keystrokes on touchscreen keyboards of mobile phones. Liu et al. [22] and Maiti et al. [24] proposed approaches for keystroke inference attacks targeting external QWERTY keyboards using motion data collected from a victim’s smartwatch, achieving word-level accuracy of over 60%. Marquardt et al. [26] introduced (sp)iPhone, where accelerometer data from a smartphone placed near a physical keyboard could be used to infer keystrokes. For their attack, they took advantage of the vibrations generated by the keystrokes on the keyboard and transmitted through the table to the smartphone’s accelerometer.

Our review of the literature highlights key gaps in existing research, motivating us to explore a new attack surface for keystroke inference. Previous studies relied primarily on stationary recording devices, such as mobile phones or dedicated microphones, or on social engineering to obtain labeled victim data [11], which makes practical attacks challenging. Moreover, the unpredictable positioning of a victim’s mobile phone in proximity to a keyboard diminishes the

practicality of such attacks. In contrast, wireless headphones are commonly worn in a predictable position on the head, even when not actively used, providing a promising and consistent opportunity for eavesdropping. Table 2 contrasts the key assumptions of these attack modalities. However, challenges such as variability in acoustic data caused by head movements and weaker vibration signals in headphones compared to devices in direct contact with the keyboard still persist. We hypothesize that combining acoustic and motion sensor data from headphones can overcome the above challenges and improve inference accuracy, and we aim to validate this through a multi-sensor framework.

Table 2. Comparison of attack assumptions across device types.

	Smartphone	Smartwatch	Headphones (<i>OverHear</i>)
Sensor placement	Near keyboard (variable)	Single wrist	Both sides of head (fixed)
Sensor separation	Limited (single device)	Limited (compact form factor)	Wide (ear-to-ear)
Placement predictability	Low (varies per session)	Moderate (worn on wrist)	High (consistent on-head position)

3 Adversary Model

3.1 System Model

We consider the setup of a user typing on a physical keyboard while wearing headphones that are equipped with:

1. **Microphones:** Microphones on each headphone earpiece, typically used to capture human audio input (e.g., voice or speech commands) or to cancel active noise (in the environment). Depending on the headphone model, each earpiece could be embedded with multiple high-definition microphones.
2. **Motion or IMU Sensors:** Accelerometers (and gyroscopes) to detect user gestures, support wearing detection, and enable features such as spatial audio or head-tracking.

During normal usage, these headphones communicate with a manufacturer-provided companion application on a smartphone or computer, which may legitimately access the headphone’s sensors to enhance audio features or provide gesture-based functionality.

3.2 Threat Model

We assume an *adversary* whose goal is to infer potentially sensitive information typed by a user on an external peripheral (e.g., keyboard).

Adversary’s Capabilities.

- **Sensor Access:** The adversary can covertly access and record microphone and accelerometer data from the user’s headphones. This could be done, for instance, through a malicious or compromised companion app that grants broad (but seemingly legitimate) permissions to read the data of the headphone sensors.
- **Remote Processing:** The adversary can offload sensor data to a remote server and run inference algorithms to predict keystrokes and reconstruct text.
- **No Direct Phone/Computer Microphone Access:** We do *not* assume that the adversary can directly access the microphone of the device paired with the headphones (e.g., a smartphone or computer). Modern OS permission models typically restrict microphone access without explicit user permission. Instead, we focus on headphones as the side-channel, which are granted access without explicit user consent to its sensor data.

Headphones are widely used and worn for extended periods, often even when no audio is actively playing. In particular, modern headphones commonly provide active noise cancellation (ANC), which encourages users to keep wearing the device for ambient noise suppression alone. Our participant survey (Appendix E) corroborates this: 52.94% of respondents reported *often* typing on a keyboard while wearing headphones or earbuds, and 17.65% reported doing so *always*, indicating that the attack window is not limited to active media playback.

Threat Landscape. Headphones are widely used and worn for extended periods, often even when no audio is actively playing. In particular, modern headphones commonly provide active noise cancellation (ANC), which encourages users to keep wearing the device for ambient noise suppression alone. Our participant survey (Appendix E) corroborates this: 52.94% of respondents reported *often* typing on a keyboard while wearing headphones or earbuds, and 17.65% reported doing so *always*, indicating that the attack window is not limited to active media playback. Unlike direct microphone access on a smartphone, which raises user suspicion (e.g., explicit notification in modern OSes), headphone sensor access can remain unnoticed under the guise of legitimate features (e.g., noise cancellation tuning, activity tracking, or equalizer settings). Unlike accessing the microphone of a mobile phone or computer, which often raises significant suspicion due to OS-level permission models, accessing headphone sensor data through APIs such as Apple’s `CMHeadphoneMotionManager` and `AVAudioRecorder` [5,6] is considerably less conspicuous.

Further exacerbating this threat is the advent of *standalone* headphones, such as STREAMZ Voice Controlled Headphones [35]. Operating independently of paired smartphones or computers, these devices bypass many of the traditional OS-level restrictions on data collection and permissions. Such devices are

capable of locally capturing and processing audio or motion data, then covertly transmitting it to attacker-controlled servers without ever engaging the user’s host device in a meaningful permission check.

Real-world incidents illustrate that such scenarios are not merely theoretical. There have been lawsuits [1] and documented cases [9, 30, 37, 39] involving popular headphone and smart-device manufacturers accused of surreptitiously collecting user data. These examples not only validate the feasibility of covertly harvesting sensitive information but also highlight the growing risk posed by modern peripheral devices. In the context of keystroke inference, motion data from accelerometers paired with headphone-based audio recordings offers attackers a potent, multi-modal avenue to learn user typing behavior; even circumventing stricter permission regimes on the phone or computer [17].

4 Design and Implementation

In this section, we outline the design of our *OverHear* inference framework and discuss the specifics of its implementation.

4.1 Inference Framework

Figure 1 illustrates the high-level architecture of *OverHear*. The attack begins by using a custom app, masquerading as the target’s headphone companion app, to covertly capture the victim’s accelerometer and microphone (audio) data during his/her typing activities. These raw data are transmitted to a remote server, where they are processed to identify the type of keyboard used by the victim using a custom-designed keyboard identification technique (see Sect. 4.7). The audio and motion sensor data streams captured through victim’s headphones go through pre-processing and segmentation to identify key presses (as detailed in Sect. 4). The detected key presses are then tested via a trained machine learning models to predict individual keystrokes. The sequences of keystrokes are then further processed in a word prediction module with the aid of a spelling correction algorithm to predict the closest matching words. The machine learning models in the attack are trained using data collected from a diverse set of users optimized for accuracy and generalization.

4.2 Data Pre-processing

Audio Noise Filtering. To ensure the clarity and relevance of our audio data in the context of keystrokes, we employed filtering to mitigate background noise, which typically occurs at higher frequencies distinct from those of keystrokes. Through analysis of the audio captured via our headphone setup, we discerned that keystrokes predominantly occur within the frequency range of 1200–3800 *Hz*. Consequently, a bandpass filter was tailored to retain information within this specific range while filtering out extraneous frequencies.

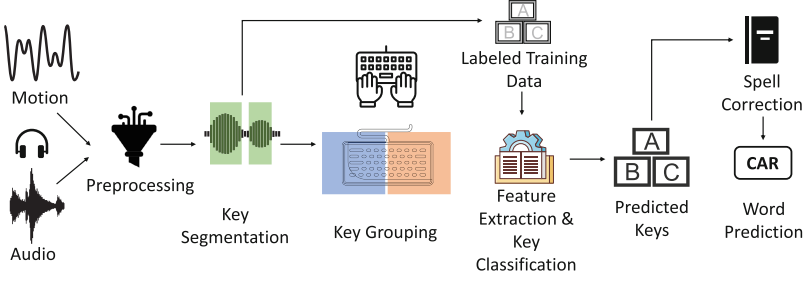


Fig. 1. *OverHear* inference framework overview.

Accelerometer Noise Filtering. The raw accelerometer data very often contains noise, primarily stemming from involuntary body movements, especially in body-worn devices. In our headphone-based attack scenario, noise arises from subtle, involuntary head movements, such as micro-adjustments or jitters, which introduce high-frequency noise; a phenomenon we empirically observed during controlled experiments. To mitigate this, we applied a low-pass filter, which effectively eliminates high-frequency noise from these movements while preserving the lower-frequency vibrations induced by key presses.

4.3 Keystroke Segmentation

In this subsection, we detail the process by which the audio signal captured by the stereo microphones of headphones is segmented into individual keystrokes. A keystroke consists of two main events: a *key hit* event and a *key release* event. An acoustic signal resulting from a keystroke is able to capture both of these events by producing two distinct peaks in the signal, each corresponding to the key hit and key release events. Figure 2 illustrates the typical acoustic feedback captured from the stereo microphones on a pair of headphones from a keyboard key press. The initial more pronounced peak represents the key hit event, while the subsequent lower peak indicates the key release event.

During the design stage, we conducted observational experiments with three participants and empirically observed that the average duration of a keystroke is approximately $80ms$. The key hit peak occurs near the beginning of the window, while the release peak appears later and is followed by lower-energy trailing content (Fig. 2). An $80ms$ window balances capturing both peaks with minimizing trailing noise and overlap with subsequent keystrokes. We use a sliding window with a size of $1ms$ (empirically determined) and calculate the energy (Eq. (1)), which is the sum of the squares of the audio amplitude values for each window and is given by the following formula:

$$E_x = \sum_n |x(n)|^2 \quad (1)$$

where E is the energy of the signal, n is the index of the audio sample and $x(n)^2$ is the square of the amplitude of the signal at the n^{th} sample. Energy is

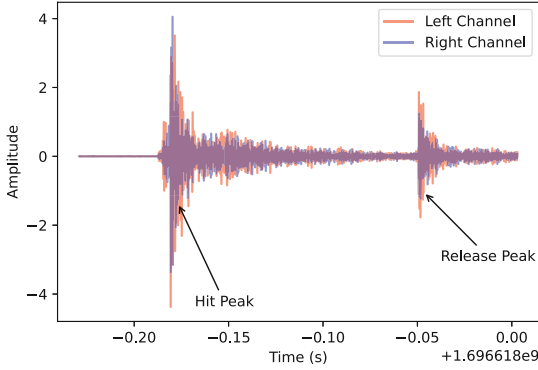


Fig. 2. Acoustic waveform of a keystroke.

a measure of the signal’s strength and is particularly useful in identifying the stronger fluctuations caused by the keystroke events. We then pass each window through an adaptive thresholding peak picking algorithm called *Music Structure Analysis Framework (MSAF)* [28] which considers a local average to pick the prominent keystroke-related peaks and to discard insignificant noise peaks. The windows with the detected peaks are then considered potential keystroke start points p_s . Consecutive start points less than 80ms are discarded (as they may belong to the same keystroke). Once a keystroke start point is identified, we define the keystroke extraction window as:

$$ks_i = (p_{si} - 5\text{ms}, p_{si} + 80\text{ms}) \quad (2)$$

where, ks_i is the i^{th} keystroke in the continuous signal, p_{si} is the start point of the keystroke, and $p_{si} + 80\text{ms}$ is the end point.

4.4 Key Group Clustering

One of the main challenges towards executing a successful keystroke inference attack is the number of potential keys and their arrangement on a keyboard, which can make the prediction task complex. This is because training a single model to distinguish between each key individually may require a vast amount of data for each key to achieve reasonable accuracy. By grouping keys and training for a group rather than individual keys, the problem can be simplified, thus making the training and inference processes more efficient. Grouping keys can improve the robustness of the predictive model by reducing the potential for confusion between similar sounding keys/keystrokes. In all cases, keys that are physically close to each other on the keyboard produce acoustically similar signals due to their proximity and the mechanics of the keyboard. By clustering keys into groups, we enable our models to become more sensitive to subtle differences within these clusters. This approach simplifies the overall task by reducing the number of direct comparisons a model must make, enhancing its ability to

accurately discern keystrokes. We first look into methods used for key grouping in previous acoustic-based keystroke inference works [21, 27] for their applicability in our attack scenario, identify challenges posed by them, and then propose techniques that suit our headphone-based inference setting.

Traditional TDoA Based Clustering. We first examine the possibility of using traditional Time Difference of Arrival (TDoA)–based key identification and key group clustering methods to identify similar keystrokes with comparable sound profiles [13]. TDoA is a commonly used technique in localization problems for both acoustic and radio-frequency signals, where it measures the difference in each signal’s arrival time across multiple receivers (e.g., microphones for acoustic signals or antennas for radio signals). The overarching principle behind TDoA is that the sound produced by a source travels through a medium (often air) and reaches one microphone before the other, if the sound source isn’t equidistant to the two microphones. This difference in time that the sound signal takes to reach each microphone is referred to as the time difference of arrival. We compute TDoA using the cross-correlation method for the 2-channel audio signal. Estimation of TDoA using cross-correlation involves finding the lag (or shift) at which two signals are most similar [8]. The cross-correlation TDoA formula is as follows:

$$\text{TDoA} = \arg \max_k \left(\sum_n S_1[n] \cdot S_2[n+k] \right) \quad (3)$$

where $S_1[n]$ is the signal at the first microphone at discrete time n , $S_2[n+k]$ is the signal at the second microphone shifted by k samples and $\arg \max_k$ indicates the shift k at which the cross-correlation is maximized (the shift where the two signals look the most similar).

However, the dynamic nature of head movements during typing tasks presented a significant challenge in maintaining consistent/reliable TDoA calculations for individual keys. Typists frequently shift their gaze, alternating between various sections of the screen and sometimes the keyboard. This constant change in head orientation made TDoA an unreliable metric to uniquely determine key positions. This is evident not only for individual users/typists, as shown in Fig. 3a, but also when considering multiple users, as depicted in Fig. 3b. The extensive spread of data points across all keys or classes further corroborates this observation.

Moreover, our attempts to identify distinct key groups based on similar TDoA values, as was done in some previous works [13, 21], proved unsuccessful. The high variability and inconsistency in the TDoA values, exacerbated by the anatomical differences and head movements of the users mentioned above, rendered the task of grouping keys with similar acoustic characteristics nearly impossible. This further highlights the unique challenges posed by our headphone-based setup compared to previous keystroke inference methods [8, 21, 27].

Energy Based Clustering. Next, we explore other techniques that could aid our framework in identifying key groups. Specifically, we explore the energy levels of keystrokes in the acoustic signal. We observed that the energy for the keystrokes towards the right of the keyboard is higher on the right audio channel

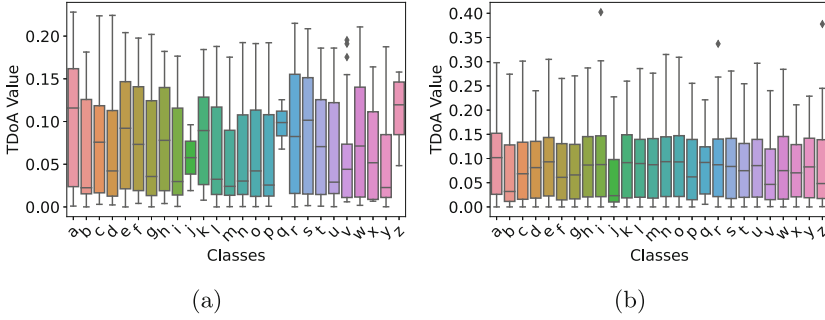


Fig. 3. Variability of TDoA (a) for a single participant, (b) across all participants.

Algorithm 1 Key Group Classification.

Require: E_R , E_R^{med} , λ

```

1: if  $E_R > E_R^{med}$  then
2:   if Prediction Probability of  $G_1 < \lambda$  then
3:     Evaluate using  $G_2, G_3$ 
4:     MaxProbability( $G_1, G_2, G_3$ )
5:   else classify using  $G_1$ 
6:   end if
7: else if  $E_R < E_R^{med}$  then
8:   if Prediction Probability of  $G_2 < \lambda$  then
9:     Evaluate using  $G_1, G_3$ 
10:    MaxProbability( $G_1, G_2, G_3$ )
11:   else classify using  $G_2$ 
12:   end if
13: else
14:   if Prediction Probability of  $G_3 < \lambda$  then
15:     Evaluate using  $G_1, G_2$ 
16:     MaxProbability( $G_1, G_2, G_3$ )
17:   else classify using  $G_3$ 
18:   end if
19: end if

```

compared to left channel, and vice versa. Under a setting similar to previous works [8, 21], where the audio recording takes place from a fixed position, such as a phone kept near the keyboard, this type of energy-based clustering can easily be used to cluster the keys into left and right groups. However, due to the constantly varying head direction changes that occur during typing tasks, which may include either looking at different parts of the screen or looking at the keyboard and then looking back at the screen, the energy differences for left and right audio channels also turned out to be not reliable and consistent.

Although the audio channels are affected by head movements and deemed unreliable for key clustering, we discovered an alternative in the form of accelerometers embedded on both sides of the headset. These accelerometers demonstrated potential in distinguishing whether a keystroke was made by the left or right hand, based on the motion feedback they recorded. To illustrate how vibrations propagated by pressing of keys affect the accelerometer, we refer to the accelerometer's coordinate system shown in Fig. 4a. The y axis corresponds to up and down head movements, which often occur when a user glances at

the keyboard and then refocuses on the screen. The x axis captures side-to-side (lateral) head movements, which may occasionally occur during typing. In contrast, the z axis is perpendicular to the sensor surface and remains the most stable. Since head movements primarily affect the x and y axes, the z axis is least influenced by motion artifacts, making it the most effective for capturing keystroke-induced vibrations belonging to the left or right hand. Figure 4b shows the accelerometer feedback captured from our prototype headphone (described in Sect. 4.8) when pressing the key ‘a’ with the left hand. The energy profile of the z axis of the left channel is clearly distinct from all other axes.

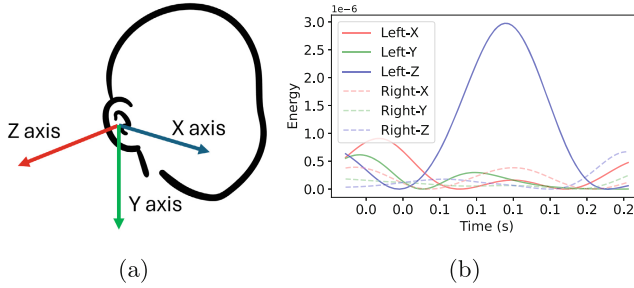


Fig. 4. (a) Accelerometer coordinate system for a headphone-wearing head, illustrating the X, Y, and Z axes, (b) accelerometer channels when pressing key ‘a’.

With the aforementioned observation on the significance of energy readings from the z -axis of the accelerometer, we proceed to quantify the distribution of energy between the left and right accelerometer channels. We achieve this by introducing an energy ratio metric. For each channel, the energy E is calculated as the sum of the squares of its samples (see Eq. (1)). The energy ratio E_R is then defined as the proportion of the left channel’s energy to the total energy of both channels, formulated as:

$$E_R = \frac{E_{\text{left}}}{E_{\text{left}} + E_{\text{right}} + \epsilon} \quad (4)$$

where ϵ is a minuscule constant introduced to prevent division by zero. This metric provides a relative energy measurement between the two channels. If a key is pressed from the left hand, the energy registered on the left accelerometer channel (of the headphones) will be higher than the right accelerometer, making the E_R closer to 1 and if it is a right-hand pressed key, E_R will be closer to 0. The energy ratio inherently normalizes the values between 0 and 1, making it easier to adapt the metric to different users. In contrast, the absolute energy values may vary based on factors such as the distance from the source of vibration/key press (due to anatomical differences), or even the user’s typing intensity.

Given the energy ratio calculated for each key press, we establish three key groups based on their spatial positioning on the keyboard and the hand predominantly used to press them.

- G_1 :** Keys predominantly pressed by the left hand, namely {a, s, d, z, x, q, w}.
- G_2 :** Keys predominantly pressed by the right hand, {o, p, k, l, n, m, i, j}.
- G_3 :** Ambiguous keys that could be pressed by either hand, namely {r, t, y, u, f, g, h, v, b, c, e}.

We observed that keys within groups G_1 and G_2 are predominantly pressed by left and right hands, respectively, across different users, especially due to their spatial positioning on the keyboard (extreme left and extreme right). However, the keys in group G_3 presented ambiguity, with the choice of hand varying from one user to another. Such variations could arise from individual typing habits or a user's inclination to favor their dominant hand. During the test phase on unseen data, the median energy ratio, E_R^{med} , is calculated for all samples for a given test user/victim. The rationale behind computing the median energy ratio is to account for the variability in key pressing intensities among different participants. Different participants may exert different pressures when pressing the keys, leading to variations in the vibrational feedback recorded by the accelerometers. Using the median, our aim was to normalize this variability and achieve an adaptive clustering mechanism. Keys with an energy ratio greater than E_R^{med} are classified as G_1 , while those with an energy ratio less than E_R^{med} are classified as G_2 . Keys with an energy ratio close to E_R^{med} , within a threshold γ , are classified as G_3 .

We then train three different classification models (as detailed in Sect. 4.5), one for each group, to predict the exact keys within each group. During this prediction, if a test keystroke is initially classified as G_1 but the prediction probability is below a certain threshold λ , the test keystroke is then also evaluated by the models for G_2 and G_3 . The final prediction is chosen from the model that yields the highest prediction probability (see Algorithm 1).

4.5 Feature Extraction and Model Training

After the key clustering step, we then investigate acoustic based features which could be used to identify individual keys. One such set of features include the *Mel Frequency Cepstral Coefficients (MFCC)*, which have been widely used in the field of speech and audio signal processing, particularly for applications such as speech recognition and speaker identification [2]. However, more recently, MFCC-based features have been used in other acoustic-related applications such as keystroke recognition and acoustic activity recognition [21, 32]. The process generally involves the following steps: (i) First, the acoustic signal is divided into fixed-sized frames, and for each frame *Fast Fourier Transform (FFT)* is applied to the acoustic signal to calculate the power spectrum. (ii) Then, *Mel Filter Bank* is applied on the power spectrum computed for each frame. The Mel Filter bank is a set of 20–40 (usually) triangular filters that are spaced according to the Mel scale, which approximates the human ear's response more closely than the linearly spaced frequency bands. This process converts the frequency power spectrum into the Mel spectrum [21]. For our inference framework, we empirically determined the optimal number of Mel Frequency Cepstral Coefficients

(MFCC), extracting 14 coefficients for each audio channel (left and right). To capture the variability and characteristics of these coefficients, we computed several statistical measures: mean, standard deviation, skewness, maximum value, median, and minimum value. This resulted in $14 \times 6 = 84$ features for each channel. Thus, by combining both channels, we derived a total of 168 features. In addition to the MFCC features, we also included *Root Mean Square Energy (RMSE)* [41] of each keystroke per channel, bringing the total feature count to 170. Building on this, we tested several models including *Random Forest classifier*, *Decision Tree Classifier* and several *Deep Neural Networks* for our analysis. To optimize its performance, we utilized a *Grid Search Cross-Validation* approach for hyperparameter tuning using a hold-out dataset. Additional details on the models used appear in Appendix B. As detailed in Sect. 4.4, we utilize three distinct multi-class classification models corresponding to the key groups G_1 , G_2 , and G_3 . Each classifier is specifically trained to identify keystrokes within its designated group. Training and testing were carried out in a *Leave-One-Out Cross-Validation (LOOCV)* manner, ensuring that a test/victim participant was excluded in each iteration. The LOOCV setup reflects a realistic attack scenario in which the adversary does not possess labeled data from the specific target user.

4.6 Word Prediction

After predicting individual keystrokes, we enhance the effectiveness of the attack by predicting the possible word formed by the inferred keystrokes. We explore two approaches for word prediction. The first is a *naive dictionary-based* method, where sequences derived from the top- k predicted keys (i.e., the correct key is among the k top predicted keys from the aforementioned key prediction models) are cross-referenced with a predefined dictionary. If a sequence matches, it is considered a valid word. However, this method struggles with inaccuracies such as missing, incorrect, or extra keystrokes. The second approach uses the *SymSpell* algorithm [15], a spelling correction method that precomputes possible spelling errors for every word in its dictionary up to a given edit distance. This allows rapid, memory-efficient word predictions and corrections, which is ideal for handling inaccuracies in keystroke predictions. The procedure (see Algorithm 2) begins by initializing the SymSpell library and loading a frequency dictionary. It then generates all possible word combinations from the top- k key predictions, checks these against the dictionary, and returns the top- w most probable corrections based on word frequency. The rationale is that words used more frequently are more likely to be the intended word in the case of a typo.

4.7 Keyboard Type Inference

A preliminary step for an attacker aiming to execute a keystroke inference attack is to infer the type of keyboard the victim employs. Given the distinct acoustic signatures produced by different types of keyboards, such as mechanical or membrane, understanding the type of keyboard (used by the victim) can pave

Algorithm 2 Word Prediction with SymSpell.

Require: *predictions*: top-k letter predictions
Require: *topK*: number of words to return

```

1: function SPELLCORRECTION(predictions, topK)
2:   Initialize sym_spell and load dictionary
3:   possible_combinations  $\leftarrow$  GENERATECOMBINATIONS(predictions)
4:   Initialize predicted_words_with_counts as empty list
5:   for each input_term in possible_combinations do
6:     Get suggestions for input_term from sym_spell
7:     Append unique suggestions to predicted_words_with_counts
8:   end for
9:   Sort predicted_words_with_counts by word frequency return First top - w words from
   predicted_words_with_counts
10: end function

```

the way for a more targeted and effective attack. In our study, we collected data from two different brands for each of the keyboard categories: **K1**: mechanical, **K2**: membrane, and **K3**: laptop-based membrane keyboards. The rationale behind using two models for each category was to introduce a level of complexity to the inference task. If the model were trained solely on data from a single brand for each category, it would trivially classify that brand during testing. Our objective with keyboard-type inference is to generalize across brands and variations within each category, ensuring the model can identify the overarching category to which they belong. For the purpose of keyboard-type inference, we segment the keyboard input audio data into 30-second windows, extracting 6 MFCC features and RMSE (Eq. (1)) for each segment. We observed during our experiments that while keystroke inference demands a much more detailed feature set, keyboard type inference can be effectively achieved with just these 6 MFCC features. Subsequently, we employ a *multi-class logistic regression* model trained on these data to predict the keyboard type. We tested the following keyboards in our keyboard-type inference experiment:

- K1**: Monoprice MP810 (red switches) and Aukey KMG12 (blue switches).
- K2**: Logitech K120 and Dynex DX-PNC2019.
- K3**: Tecknet Ultra Slim Compact and the keyboard of the HP Envy 15" laptop.

4.8 Experimental Setup

Due to the absence of published publicly-available APIs for accelerometer data extraction in the current generation of commercial headphones, and the limitations associated with the few available APIs such as those from Apple, we devised a custom setup to comprehensively evaluate all the parameters and settings of *OverHear*. However, to test the efficacy of *OverHear* on commercially available hardware, we also conducted some limited experiments using Apple AirPods.

Our primary experimental setup comprises of a Raspberry Pi, and a 3D-printed over-the-ear headphone prototype (see Fig. 12a). To capture audio, we

equipped each earpiece with an Adafruit I2S MEMS Microphone and a MPU-6500 accelerometer to record motion data. The audio was sampled at a maximum of $96kHz$ while accelerometer was sampled at a maximum of $500Hz$. The microphones and accelerometers were connected to the Raspberry Pi via the GPIO interface, and a Python script was used to record the data from each microphone/sensor. For our tests involving commercially available hardware, we employed a pair of Apple AirPods Pro (2nd Generation) [4] connected to an iPhone SE. We developed an iOS application using Swift, which utilized the `CMHeadphoneMotionManager` API [6] to collect motion data (note: this API does not allow specifying a sampling rate) and captured audio data at a sampling rate of $44kHz$, the maximum allowed. The evaluation of the data processing and inference framework was performed on an Ubuntu 22.04 virtual machine with 32GB of memory and 32 cores using Python 3.10.

Our keystroke inference experiments included a diverse range of keyboards to ensure a comprehensive evaluation. We used an AUKEY KMG12, a full-sized mechanical keyboard (104 keys) to represent **K1** category. For the **K2** category, we utilized a Logitech K120 membrane type, full-sized keyboard model. To closely replicate a keyboard of modern laptops a Tecknet Ultra Slim Compact keyboard (68 keys), was used (representing the **K3** category). This diversity in keyboard types allowed us to assess the robustness and adaptability of our framework across different tactile feedback mechanisms and form factors. We collected typing data from 17 participants (plus 5 additional participants for the AirPods setup), under an IRB-approved protocol; full data-collection details are provided in the Appendix A.1.

5 Evaluation

Next, we present the metrics used and evaluation results for *OverHear* under a variety of different experimental settings and conditions.

5.1 Metrics

Precision and Recall. Precision measures the number of correctly predicted keystroke segments out of the total predicted keystroke segments, while recall (or sensitivity) calculates the number of correctly predicted keystroke segments out of the actual keystroke segments. We also use precision and recall to measure the prediction performance of our keyboard-type inference module.

Top- k_{key} Accuracy. This metric evaluates the accuracy of the top- k key predictions. Specifically, if the true label is within the top- k predicted labels, then the prediction is considered correct. We used top- k_{key} accuracy for assessing the performance of *OverHear* framework at the individual key prediction level.

Top- k_{word} Accuracy. The accuracy of the top- k word predictions. If the true word is within the top- k predicted words, the prediction is deemed accurate.

5.2 Keyboard Type Inference

Across all keyboard type categories, our keyboard type identification model demonstrated robust performance, consistently achieving an accuracy exceeding 0.95 when training data includes data from the same brand of keyboard. However, when the type-inference model is trained using one keyboard brand for each category, the performance slightly degrades. Nevertheless, except for **K2** category of keyboards, both other categories (see Table 3) demonstrated a recall greater than 0.95. This highlights our model’s efficacy in accurately inferring the keyboard type a victim is using.

Table 3. Keyboard type inference performance.

Keyboard Type	Precision	Recall
Mechanical (K1)	0.86	0.96
Membrane Type 1 (K2)	0.98	0.76
Membrane Type 2 (K3)	0.88	0.99

5.3 Keystroke Detection

Our keystroke segmentation algorithm exhibited consistent performance across various types of keyboards (see Fig. 5a). For keyboard type **K1**, the precision and recall were measured at 0.80 ($\sigma = 0.05$ and $\sigma = 0.08$, respectively). For **K2**, the precision was 0.78 ($\sigma = 0.07$) and the recall was 0.77 ($\sigma = 0.07$). For **K3**, we observed a precision of 0.75 ($\sigma = 0.08$) and a recall of 0.80 ($\sigma = 0.06$). Figure 5b illustrates the precision and recall of our keystroke detection algorithm (detailed in Sect. 4.3). Notably, the algorithm’s performance remains consistent even at higher typing speeds. Although these results indicate stability in performance, there are inherent challenges that contribute to the slightly lower accuracy. One primary challenge arises when the keys are pressed quickly in succession. In particular, with adept typists, the acoustic energy from one key press can overlap with that of the subsequent key, complicating the distinction between individual keystrokes. In addition, the unique typing dynamics of each individual introduces variability. Some users exert varying force on the keys, while others occasionally press two keys nearly simultaneously. These issues add layers of complexity to the detection process.

5.4 Overall Performance

As mentioned earlier, we evaluated the performance of *OverHear* across three keyboard type categories, **K1**, **K2**, **K3**. The results revealed that, **K1**, achieved the highest top-5_{key} accuracy of 0.77 and a top-10_{key} accuracy of 0.88. In comparison, **K2**, recorded a top-5_{key} accuracy of 0.58, and **K3**, had a top-5_{key} accuracy of 0.53. Mechanical keyboards (category **K1**) tend to produce near distinct

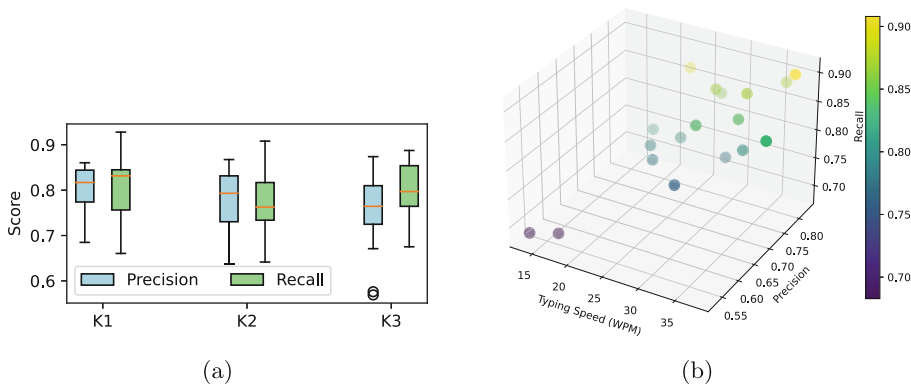


Fig. 5. (a) Keystroke detection vs keyboard types, (b) Precision and recall vs. typing speed (WPM) for **K1**.

tactile feedback and sound profiles for each key press. This unique acoustic signature for each key can make it easier for the system to differentiate between keystrokes, leading to higher accuracy as observed in our results. Although larger external membrane keyboards (category **K2**) too produce some amount of tactile feedback, the sound profiles might not be as distinct as those of mechanical keyboards. The smaller membrane keyboards (type **K3**), which closely resembles laptop keyboards, typically have keys closer together, leading to overlapping or less distinct sound profiles, especially when the keys are pressed in rapid succession. Additionally, the build and material of such keyboards might further dampen the sound, making it harder to accurately infer keystrokes.

To evaluate the efficacy of our clustering algorithm, we compared the accuracy of our model with and without the clustering approach. As presented in Table 4, the clustering algorithm considerably improved the accuracy across all keyboard types. For mechanical keyboard type, **K1**, the top-5_{key} accuracy improved from 0.59 to 0.77. Similarly, for the membrane type, **K2**, there was a noticeable increase from 0.41 to 0.58. The membrane type, **K3**, also saw an improvement in accuracy, with top-5_{key} accuracy rising from 0.37 to 0.53. These results show the effectiveness of the accelerometer-based clustering algorithm in refining the keystroke inference, making it an important component of *OverHear*.

Table 4. Top-5_{key} accuracy with and without clustering.

Keyboard Type	w/o Clustering	with Clustering
K1	0.59	0.77
K2	0.41	0.58
K3	0.37	0.53

In our evaluations, the Random Forest classifier consistently outperformed other models across all keyboard types (see Fig. 6a). Specifically, for **K1**, it achieved a top-5_{key} accuracy of approximately 0.77, while for **K2** and **K3**, the accuracies were 0.57 and 0.53, respectively. In contrast, the Decision Tree classifier managed a top-5_{key} accuracy of 0.57 for **K1** and around 0.33 for both **K2** and **K3**. The DNN models were the least effective, with accuracies falling below 0.15 for all keyboard categories likely because DNNs require substantial amount of data for effective training, beyond the dataset that we collected. The CNN-LSTM hybrid model showed a slight improvement, achieving accuracies above 0.25 for certain characters, but its overall accuracy remained low. The Random Forest classifier outperformed other models, likely due to its ensemble nature, effectively capturing complex patterns without overfitting. Moving forward, exploring alternative models such as single- and few-shot learning techniques could potentially offer more robust solutions, especially when training data is limited [42].

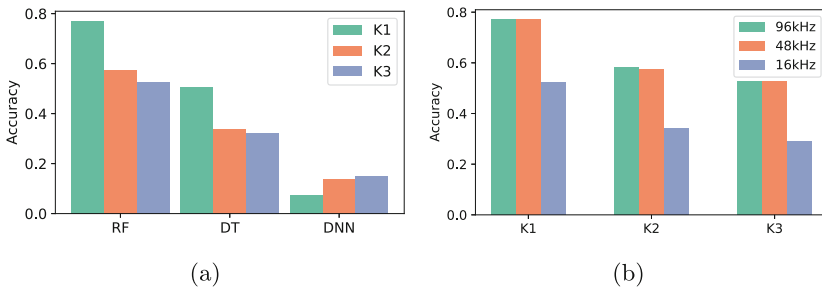


Fig. 6. Top-5_{key} accuracy for different (a) models, (b) sampling rates.

5.5 Sampling Rates

In our experiments, we explored the impact of different sampling rates on the performance of *OverHear* (see Fig. 6b), starting from our default rate of 96kHz for the audio signal. Our findings indicate that the performance at 48kHz is nearly on par with that at 96kHz. However, when the sampling rate is reduced to 16kHz, we observed a considerable degradation in performance. This indicates that our framework can work fairly well even at lower sampling rates.

5.6 Ambient Noise

We evaluated the robustness of *OverHear* under various ambient noise conditions. These noise profiles were separately collected in three different environments and then combined with keystroke data from all participants. In a university cafeteria setting, the environment was filled with ambient sounds of people

eating, working on their laptops, and background music. The open office space had 2–3 people working nearby on computers, accompanied by the typical sounds of typing, mouse clicks, and the occasional ringing of a mobile phone. In contrast, the closed office space provided a quiet environment with minimal background noise. From the results (see Fig. 7a), it is evident that accuracy is highest in quieter environments, such as a closed office, and decreases with increasing ambient noise, with the cafeteria setting being the most challenging, with accuracies dropping below 0.45. This trend is consistent across all three types of keyboard. This shows that our framework works well in quieter (ideal) environments, but the accuracy is reasonable even in noisier (less ideal) environments.

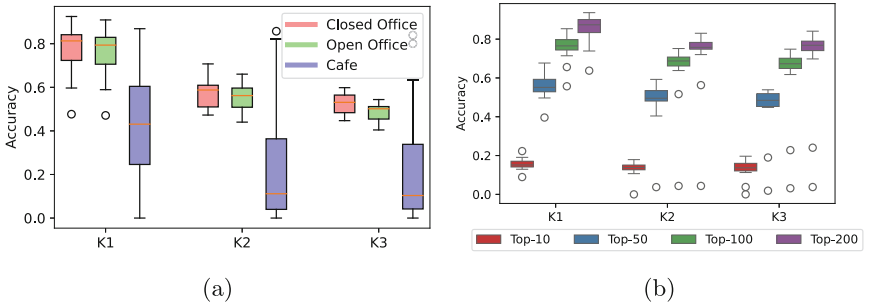


Fig. 7. (a) Top-5_{key} accuracy vs. types of ambient noise, (b) Top-*k*_{word} accuracies.

5.7 Word Prediction

Our word prediction technique, leveraging the *SymSpell* library, as seen in Fig. 7b, demonstrated considerable efficacy, achieving top-50_{word} accuracies nearing 0.50 across all keyboard categories. Further, **K1** reached a 0.76 accuracy for top-100_{word} predictions, while **K2** and **K3** closely followed with 0.71 and 0.70, respectively. For **K1**, five out of six, participants achieved an accuracy exceeding 0.60 for top-50_{word} predictions, with only one participant falling below the 0.40 mark. In the case of **K2** and **K3**, barring one outlier in each category, all participants consistently achieved around the 0.50 accuracy level for top-50_{word} predictions. In contrast, the naive dictionary-based approach, which relies solely on exact matches, lagged behind. Its limited adaptability to variations in keystroke data meant that it consistently registered accuracies below 0.4 for all keyboard types. These results can be attributed to SymSpell’s ability to efficiently handle typographical errors, which aids in more accurate word suggestions, especially in the presence of potentially inaccurate key predictions and extra/missing keystrokes.

5.8 Effect of Typing Speeds

Encompassing keystroke segmentation and subsequent processing, our framework demonstrates consistent key prediction performance across varying typing speeds (see Fig. 8). Although most of the participants achieved consistent accuracy, an exception was observed in a participant with a typing speed of approximately 35 words per minute (WPM), who registered an accuracy below 0.5. Despite this outlier, the overall robustness of the framework across varied typing speeds is evident, highlighting its adaptability to diverse user behaviors.

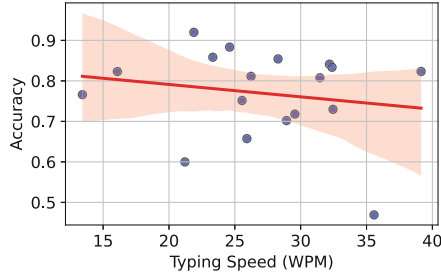


Fig. 8. Typing speeds vs. key-level accuracy.

5.9 Factors Affecting Accuracy

In our analysis of the factors that can affect the accuracy of *OverHear*, several patterns emerged. For the key group G_2 , no major misclassification patterns were observed. However, there were instances where the key ‘k’ is misinterpreted as ‘i’, and ‘m’ is confused with ‘n’. In the G_3 group, the keys ‘j’ and ‘h’ as well as ‘y’ and ‘r’ are often interchanged. Furthermore, the keys ‘q’ and ‘w’ consistently exhibit lower accuracy rates. In G_1 we observed that the key ‘z’ is frequently misclassified as ‘x’, and ‘x’ is often mistaken for ‘s’ (see Figs. 9a and 9b). These confusion patterns can be mainly attributed to the spatial proximity of these keys on the keyboard, leading to similar acoustic profiles that can be challenging to distinguish. Figure 10 visualizes this relationship between the frequency of misclassifications and the Euclidean distance between the ground truth and predicted keys on a QWERTY keyboard, classified into our three key groups: G_1 , G_2 , and G_3 . A prominent observation from the plot is that misclassifications are more frequent for keys that are closer in distance, particularly for the G_3 group. This suggests that keys in the G_3 group are often confused with their immediate neighbors on the keyboard. The plot shows the inherent challenge in distinguishing between keystrokes of adjacent keys, emphasizing the spatial aspect of the misclassification problem on a physical keyboard layout.

Furthermore, certain users possess the ability to type without glancing at the keyboard, maintaining a steady gaze on the screen. This consistent posture

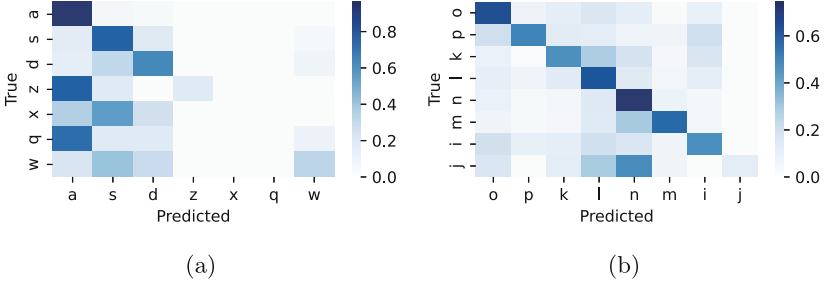


Fig. 9. Confusion matrix for (a) group G_1 keys, (b) group G_2 keys.

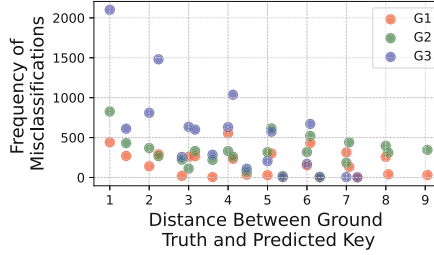


Fig. 10. Frequency of misclassifications vs. key distance.

ensures minimal head movement, leading to relatively stable acoustic profiles. In contrast, users who frequently look down at the keyboard introduce regular vertical head movements. These continuous and pronounced shifts could also influence the acoustic signatures, challenging the models employed by our framework to accurately distinguish and classify keystrokes.

To get a better understanding of such head movements and their potential correlation to the accuracy of *OverHear*, we analyzed the frequency spectrum of the gyroscope data (collected alongside accelerometer data in our custom setup). Particularly, we looked at the median frequency of each participant, which can be considered as the frequency below which 50% of the power of the signal lies. A higher median frequency in gyroscope data typically indicates more rapid changes in the signal, which can be interpreted as more intense or faster head movements. As it can be seen in Fig. 11, the three participants with the higher median gyroscope frequency showed the lowest top-5_{key} accuracies with values below 0.70. This could be intuitively attributed to the fact that more intense head movements might introduce more noise or variability in the audio/accelerometer data, making keystroke inference more challenging.

5.10 Performance on Commercial Devices

We tested our inference framework using a pair of Apple AirPods and focused on keyboard type **K1**. However, the API provided by iOS did not deliver a con-

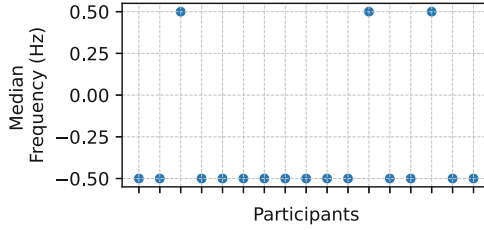


Fig. 11. Participants vs. their median gyroscope frequency.

sistent data stream. Significant gaps in accelerometer data collection affected our ability to utilize these data effectively. Consequently, we were forced to rely solely on audio data for our analysis. Despite these limitations, our framework, tested only with audio data, achieved a top-5_{key} prediction accuracy of 56% and a top-10_{key} prediction accuracy of 80%. These results are comparable (despite being slightly lower) to the performance of our framework when tested without the clustering step using our custom hardware setup (see Table 4). These results underscore the potential of audio-based keystroke inference using smart headphones, even under limiting conditions and despite the noise-cancelling (in the microphones) features inherent in these commercial headphones’ audio recording functionality. Beyond AirPods, commercial headphones with built-in accelerometers (e.g., Samsung Galaxy Buds, Google Pixel Buds) lack official APIs for continuous sensor streams, making larger-scale testing impractical at present. Nevertheless, Apple holds a significant market share (over 20% of wireless headphones [34]), indicating that our findings remain particularly relevant.

6 Discussion

Attack Impact. The widespread use of headphones, such as Apple AirPods [4], equipped with advanced microphones and motion sensors, underscores the potential reach of our keystroke inference attack. These sensors enhance the user experience, but also make the devices susceptible to *OverHear*. Our results show high accuracy on both mechanical and membrane keyboards, confirming the attack’s feasibility in real-world scenarios. Specifically, mechanical keyboards have seen a resurgence in popularity over the past decade, especially among certain demographics, such as the gaming community, technology enthusiasts, and professional typists [36]. Given the projected Compound Annual Growth Rate (CAGR) of 6.79% from 2022–2027, it is evident that this user base is not only substantial but also expanding [36]. Our inference framework, which demonstrates increased efficacy for mechanical keyboards, poses a notable threat to this growing user base. A question asked during the participant survey (Appendix E) further provides a preliminary idea of the threat landscape, with 12% of the respondents using mechanical keyboards and 47% using membrane keyboards.

Mitigations. Noise-canceling features included in most modern headphones offer a promising avenue to counteract the threat of acoustic keystroke infer-

ence. Originally designed to minimize ambient sounds, this can be further optimized to specifically target and suppress the key-press sounds. While it is challenging to completely mute the sound of keystrokes, integrating such targeted noise-canceling features to the headphones can significantly degrade the quality of captured keystroke sounds, thereby reducing the effectiveness of inference attacks. Using quieter keyboards, such as scissor keyboards [20], commonly found on laptops, can also mitigate such attacks due to their minimal key-press noise from their scissor-like hinge structure. In addition, implementing system-level monitoring services on paired devices, such as smartphones or computers, can help. These services would track the volume and frequency of data transmission by apps; Any unusual activity, such as excessive data offloading by a companion headphone app, could then be flagged for review.

Limitations. Despite promising results, our keystroke inference framework using headphones has limitations. First, it requires a malicious headphone companion app to covertly collect sensor data, which may not always be trivial to install given the reliance on the victim deciding to install it. Second, although our user study covers three representative keyboard categories, only a single keyboard was tested per category, limiting generalizability across different models within each category. In addition, accurately predicting complex passwords (mixing alphanumeric characters, symbols, and cases) remains challenging, especially for unconventional passwords. Third, testing on commercial headphones has been challenging due to the lack of official APIs to obtain continuous sensor data. Furthermore, performance degrades considerably in very noisy environments (as shown in Sect. 5.6). Lastly, although our experiments primarily focused on the 26 letters of the alphabet, it is important to note that our method can be extended to include other characters such as numbers and symbols since our framework relies on key-based sound profiles and their spatial positions.

7 Conclusion

In this study, we presented *OverHear*, a framework that adeptly exploits both acoustic and accelerometer data from smart headphones to infer keystrokes, achieving a top-5 key accuracy close to 80% for mechanical keyboards and 60% for membrane keyboards. In addition, we were able to achieve top-100 word accuracy of over 70% for all keyboard categories. Although our results highlight the vulnerabilities introduced by modern headphones in real-world scenarios, they also emphasize the importance of understanding and addressing these emerging security challenges.

Acknowledgments. This study was funded by U.S. National Science Foundation (NSF) under award #1943351.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

A Data Collection Setup

Figure 12 shows a participant using our custom and commercial hardware setups.

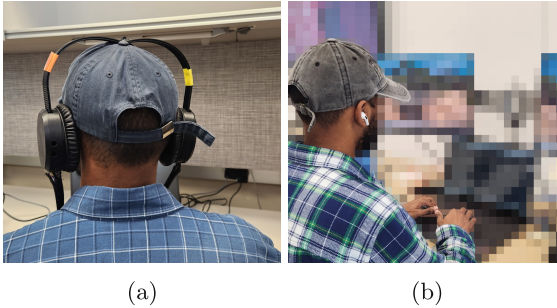


Fig. 12. A user wearing (a) our custom-built headphone setup, (b) AirPods.

A.1 Data Collection

We recruited 17 participants in the age range of 18–38 (8 female, 9 male), to collect typing data using our custom prototype pair of headphones. Participants participated in two experiments across three types of keyboards while wearing our custom headset (as described in Sect. 4.8). The experiments were conducted inside a closed office space. For the first experiment, they typed individual English alphabets displayed on the screen one at a time, where each alphabet is repeated five times at random. The second experiment involved them typing 300 English words from a list of 5000 most frequent words, with the number of letters (in each word) ranging from three to seven [43]. Participants participated in typing tasks using a 24-inch computer monitor. To ensure comfort and natural typing posture, a height adjustable chair was provided, allowing them to choose the level of seating they found most comfortable. We did not impose specific typing techniques on the participants; instead, they were encouraged to type in a manner consistent with their daily habits. For the additional commercial headphone setup using AirPods, we recruited 5 participants who performed the same aforementioned experiments on keyboard **K1**. Participants also completed a short survey at the end of the experiments, which included questions related to their typing behaviors and headphone usage. Some interesting insights obtained through the survey are given in Appendix E. Our data collection procedures were approved by our institution’s Institutional Review Board (IRB).

B Model Details

The *Random Forest Classifier* was configured with balanced class weights to counteract the uneven distribution of class frequencies, enhancing fairness and

accuracy in the model’s predictions. It was set with a maximum depth of 20, and used ‘sqrt’ for the number of features considered when looking for the best split. The *Decision Tree Classifier* employed a gini impurity criterion with balanced class weights for splits, and a moderate tree depth of 10 to prevent overfitting, alongside tuned parameters for minimum samples per leaf and split to ensure robust nodes. For our baseline *DNN*, a sequential architecture was implemented beginning with an input layer of 256 neurons, followed by a 50% dropout rate to mitigate overfitting. The output layer uses a softmax activation function to provide a probability distribution across the classes, also addressing class imbalance by weighting the loss function accordingly. This model was compiled with an Adam optimizer and categorical cross-entropy loss function, which is particularly suited for multi-class key classification where class imbalance is a concern.

We also explored more advanced deep learning models. First, we adopted the CNN model proposed by Giallanza et al. [16] for acoustic keystroke inference. The model consists of two convolutional layers with batch normalization and max pooling to extract spatial features while reducing dimensionality. The first convolutional layer applies 64 filters (3-sized kernel), followed by batch normalization and max pooling to stabilize training and prevent overfitting. The second convolutional layer increases the filter count to 128, enabling deeper feature extraction. The output is then flattened and passed through dense layers (256 and 128 neurons, ReLU activation) with 50% dropout before classification via a softmax layer. The model is trained using Adam optimization with a categorical cross-entropy loss function, suitable for multi-class classification. Beyond CNNs, we explored a CNN-LSTM hybrid model to capture both spatial and temporal dependencies in keystroke signals. The architecture mirrors the CNN model but integrates an LSTM layer (128 units) after feature extraction to learn temporal relationships, such as press-release timing patterns. The LSTM output is processed through dense layers (256 neurons, dropout 50%) before classification. Like the CNN, it is optimized using Adam and categorical cross-entropy loss.

C Word Prediction Under Noise

Figure 13 illustrates the word-prediction performance in a cafe ambient noise setting. Despite the key prediction accuracy being notably lower than in quieter environments, top-100_{word} predictions still approached an accuracy close to 0.60.

D Participant Survey

1. What is your age?
2. What is your dominant hand?
Right, Left
3. Do you currently own any of the following digital devices?
 - Yes, No - Bluetooth Over-the-Ear Headsets
 - Yes, No - Bluetooth Earbuds

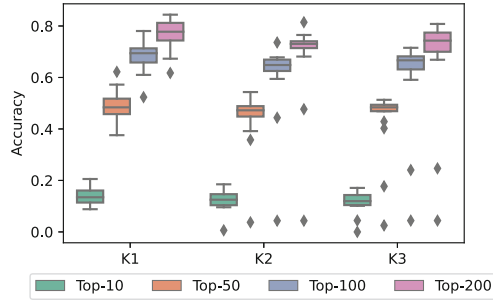


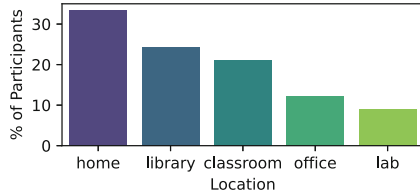
Fig. 13. Word-level top-w accuracy under cafeteria noise.

4. How often do you type on a computer keyboard while wearing a pair of headsets/earbuds?
Never, Rarely, Sometimes, Often, Always
5. How many hours a day do you perform typing tasks in general?
6. How many hours a day do you perform typing tasks while wearing a head-set/earbuds?
7. What type of a keyboard do you own?
Membrane, Mechanical, Not Sure
8. Where do you typically perform your typing tasks?
café, library, classroom, home, other (please specify)
9. Have you installed the smartphone app that comes bundled with your head-set/earbuds?
Yes, No
10. Are you aware that modern headphones and earbuds have embedded motion sensors in them?
Yes, No

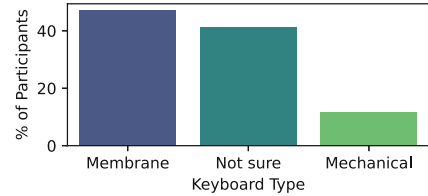
E Insights from Participant Survey

Through the responses from our participant survey, we were able to gain the below insights into the individual typing behaviors and the prevalent patterns of headphone usage among participants (Fig. 14).

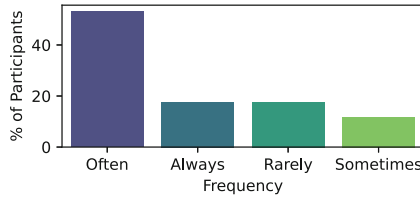
- A majority of participants (52.94%) often type on a computer keyboard while wearing headphones or earbuds, with 17.65% doing so always. On average, 11.76% type for 2.5 h per day with these devices on.
- 52.94% of participants were aware that headphones and earbuds often have embedded motion sensors.
- In terms of keyboard ownership: 47.06% owned a membrane keyboard, 11.76% owned a mechanical keyboard and 41.18% were uncertain about their keyboard type.
- As for typing locations, 33.33% of participants typically type at their home or apartment, and 24.24% prefer the library.



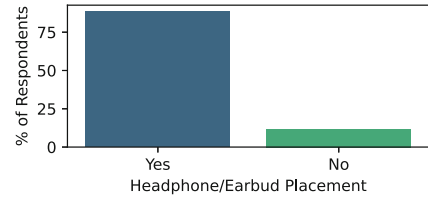
(a) Distribution of locations for typing tasks.



(b) Distribution of keyboard types owned by the participants.



(c) Frequency of typing on a computer keyboard while wearing headsets/earbuds.



(d) Distribution of participants who do/do not keep their headphones/earbuds near the computer.

Fig. 14. Survey results on typing behavior and device usage.

References

1. ABC: Lawsuit alleges Bose collecting user data without permission (2017). <https://abc7chicago.com/bose-consumer-concerns-spying-data-mining/1897655/>. Accessed 15 June 2024
2. Ali, S., Tanweer, S., Khalid, S.S., Rao, N.: Mel frequency cepstral coefficient: a review. ICIDSSD (2020)
3. Anker: Soundcore (2024). <https://apps.apple.com/us/app/soundcore/id1331876603>. Accessed 15 June 2024
4. Apple: AirPods Pro (2022). <https://www.apple.com/airpods-pro/specs/>. Accessed 15 June 2024
5. Apple: AVAudioRecorder (2024). <https://developer.apple.com/documentation/avfaudio/avaudiorecorder>. Accessed 15 June 2024
6. Apple: CMHeadphoneMotionManager (2024). <https://developer.apple.com/documentation/coremotion/cmheadphonemotionmanager>. Accessed 15 June 2024
7. Asonov, D., Agrawal, R.: Keyboard acoustic emanations. In: IEEE S&P (2004)
8. Bai, J.X., Liu, B., Song, L.: I know your keyboard input: a robust keystroke eavesdropper based-on acoustic signals. In: ACM Multimedia (2021)
9. BBC: Ring doorbell 'gives Facebook and Google user data (2020). <https://www.bbc.com/news/technology-51281476>. Accessed 15 June 2024
10. Cai, L., Chen, H.: On the practicality of motion based keystroke inference attack. In: IEEE TrustCom (2012)
11. Cecconello, S., Compagno, A., Conti, M., Lain, D., Tsudik, G.: Skype & type: keyboard eavesdropping in voice-over-IP. ACM TOPS (2019)
12. Chen, C.S.: Earpieces with gesture control (2015), uS Patent App. 13/959,109 (2015)

13. Cheng, K., Li, W., Zhang, L., Ma, X., Chen, J.: Dictionary attacks based on TDOA using a smartphone. In: IEEE IAEAC (2022)
14. Fan, X., et al.: Headfi: bringing intelligence to all headphones. In: ACM MobiCom (2021)
15. Garbe, W.: SymSpell (2012). <https://github.com/wolfgarbe/SymSpell> (2012). Accessed 15 June 2024
16. Giallanza, T., et al.: Keyboard snooping from mobile phone arrays with mixed convolutional and recurrent neural networks. IMWUT (2019)
17. Google: Android developers - permissions (2024). <https://developer.android.com/guide/topics/manifest/permission-element>. Accessed 15 June 2024
18. Google: Google pixel buds (2024). <https://play.google.com/store/apps/details?id=com.google.android.apps.wearables.maestro.companion>. Accessed 15 June 2024
19. Harrison, J., Toreini, E., Mehrnezhad, M.: A practical deep learning-based acoustic side channel attack on keyboards. In: 2023 EuroS&P (2023)
20. KBE team: What are scissor switch keyboards (2023). <https://keyboardsexpert.com/what-are-scissor-switch-keyboards/>. Accessed 15 June 2025
21. Liu, J., Wang, Y., Kar, G., Chen, Y., Yang, J., Gruteser, M.: Snooping keystrokes with mm-level audio ranging on a single phone. In: ACM MobiCom (2015)
22. Liu, X., Zhou, Z., Diao, W., Li, Z., Zhang, K.: When good becomes evil: keystroke inference with smartwatch. In: ACM CCS (2015)
23. Lu, L., et al.: KeyListener: inferring keystrokes on qwerty keyboard of touch screen through acoustic signals. In: IEEE INFOCOM (2019)
24. Maiti, A., Armbruster, O., Jadhwal, M., He, J.: Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In: ACM AsiaCCS (2016)
25. Mao, X., Rimon, N.: Method and apparatus for enhancing the generation of three-dimensional sound in headphone devices (2012), uS Patent 8,160,265 (2012)
26. Marquardt, P., Verma, A., Carter, H., Traynor, P.: (SP)iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In: ACM CCS (2011)
27. Narain, S., Sanatinia, A., Noubir, G.: Single-stroke language-agnostic keylogging using stereo-microphones and domain specific machine learning. In: ACM WiSec (2014)
28. Nieto, O., Bello, J.P.: Systematic exploration of computational music structure research. In: International Society for Music Information Retrieval Conference (2016)
29. Prest, C., Hoellwarth, Q.C.: Sports monitoring system for headphones, earbuds and/or headsets (2014), uS Patent 8,655,004 (2024)
30. Reviews, T.: Is your smart TV spying on you? All you need to know about smart TVs and your privacy (2020). <https://www.trustedreviews.com/news/smart-tv-privacy-problems-vizio-samsung-lg-sony-panasonic-2952175>. Accessed 15 June 2025
31. Röddiger, T., et al.: Sensing with earables: a systematic literature review and taxonomy of phenomena. ACM IMWUT **6**(3) (2022)
32. Roth, J., Liu, X., Ross, A., Metaxas, D.: Biometric authentication via keystroke sound. In: ICB (2013)
33. Samsung: Galaxy buds manager (2024). <https://play.google.com/store/apps/details?id=com.samsung.accessory.fridaymgr>. Accessed 15 June 2025
34. Statista: Market share of the leading true wireless stereo (TWS) vendors (2024). <https://www.statista.com/statistics/1105407/leading-tws-vendors/>. Accessed 15 June 2025
35. STREAMS: Streamz voice controlled headphones (2024). <https://www.streamzmedia.com>. Accessed 15 June 2024

36. TECHNAVIO: Mechanical keyboard market by distribution channel, type, and geography - forecast and analysis 2023–2027 (2023). <https://www.technavio.com/report/mechanical-keyboard-market-industry-analysis>. Accessed 15 June 2024
37. Temming, M.: Smartphones put your privacy at risk (2018). <https://www.commonlit.org/en/texts/smartphones-put-your-privacy-at-risk>. Accessed 15 June 2024
38. Terlizzi, J.J.: Systems and methods for noise cancellation and power management in a wireless headset (2012), uS Patent 8,285,208 (2012)
39. TheVerge: Anker finally comes clean about its Eufy security cameras (2023). <https://www.theverge.com/23573362/anker-eufy-security-camera-answers-encryption>. Accessed 15 June 2024
40. Wang, H., Lai, T.T.T., Roy Choudhury, R.: Mole: motion leaks through smartwatch sensors. In: ACM MobiCom (2015)
41. Wang, J., Ruby, R., Wang, L., Wu, K.: Accurate combined keystrokes detection using acoustic signals. In: IEEE MSN (2016)
42. Wang, Y., Yao, Q., Kwok, J.T., Ni, L.M.: Generalizing from a few examples: a survey on few-shot learning. *ACM Comput. Surv.* **53**(3) (2020)
43. Word Frequency: Introduction to the top 60,000 words in English (2023). <https://www.wordfrequency.info/intro.asp>. Accessed 15 June 2024
44. Yu, Z., Kaplan, Z., Yan, Q., Zhang, N.: Security and privacy in the emerging cyber-physical world: a survey. *IEEE Commun. Surv. Tutorials* **23**(3) (2021)
45. Zhu, T., Ma, Q., Zhang, S., Liu, Y.: Context-free attacks using keyboard acoustic emanations. In: ACM CCS (2014)