Privacy Risks of Conference Calls: Leaking Your Password through Audio Side-Channel

Alberto Zabeo   Mattia Evangelisti

zabeo | mattiaev @kth.se

January 16, 2023

Abstract

The widespread use of online conference meeting software during the COVID-19 pandemic is likely to continue as it offers convenience for many businesses and institutions. However, the data that was once transmitted verbally now travels through these additional mediums, which can pose a significant risk to confidentiality if used incorrectly. In this paper, we will investigate how these new technologies can be exploited to pose new threats to the user and propose a solution. Specifically, we aim to reveal a security threat that relies on the audio stream of a video conference. As previously demonstrated in other studies, the sound generated by a typing keyboard can contain enough information to reconstruct the typed text. However, previous works have produced results with either poor accuracy or by relying on access to the same keyboard used to collect data. In this paper, we will present a new keyboard acoustic eavesdropping attack based on sound analysis in the frequency domain that can be replicated on different keyboards.

Contents

1 Introduction

In this study, we will focus on the threat of acoustic eavesdropping, specifically examining the feasibility of reconstructing typed text from the audio stream of a video conference call with little or no prior information. Electronic devices are essential in modern life and often contain sensitive information that we work to protect through secure storage and careful sharing with trusted individuals. However, it is important to recognize that physical security is not the only type of security to consider. We must also address potential vulnerabilities in online and digital security in order to fully protect ourselves and our personal information.

Previous attacks in this area have required certain conditions in order to be successful. For example, [1], [2], [3], [4], and [5] all require the presence of a microphone controlled by the adversary near the victim, which limits the practicality of this method. [4] and [2] also require detailed profiling of the victim's typing style, while others require access to the same keyboard model used during the training phase of the project [1], [6].

The purpose of this work is to create a method for reconstructing typed text from a victim's laptop without the use of any external hardware or knowledge of the victim's specific device. We also aim to raise awareness of this often overlooked threat and suggest potential mitigations that can reduce the effectiveness of such attacks without impacting the quality of the received audio stream.

2 Related Works

Eavesdropping on keystroke sounds during conference calls has been and still is an active area of research. This section will overview the previous works conducted on the cited topic, emphasizing their differences and their similarities to our project. Asonov and Agraval were the first to publish research on keystroke recognition using neural networks trained on a specific keyboard. They showed that this method could achieve good results and explained that the reason for its effectiveness is due to the sounding board behavior of the space below the keys, which causes keys that are close in space to produce similar sounds. [1]

Other works rely on statistical properties of the spectrum, applying both supervised [5], [4], [1] and unsupervised [2], [7], [6] learning techniques.

Supervised learning approaches yield very high accuracy. However, they require many labeled samples and are highly dependent on hardware-specific and user-specific

Unsupervised learning techniques, on the other hand, have the advantage that they do not require a ground truth relying on the relative frequency of letters in the input language [2].

3 Methodology

3.1 Data collection

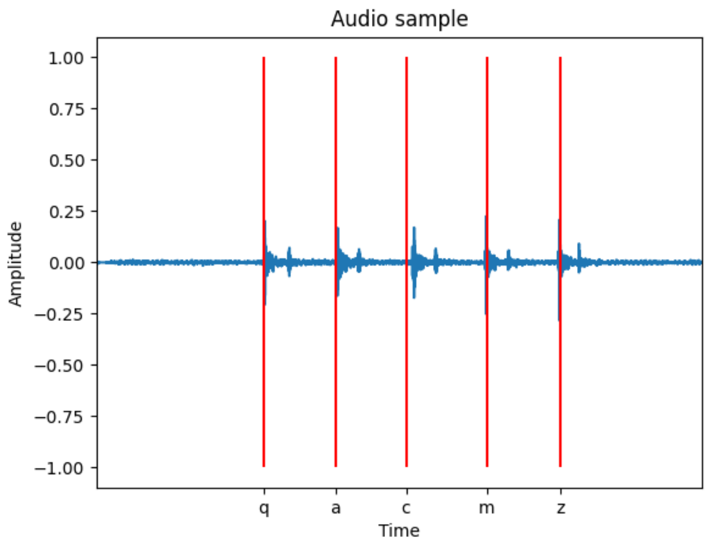

Figure 1: Audio sample from the dataset containing 5 keystrokes marked by red lines

In order to assess the effectiveness of our unsupervised learning approach and fine-tune various strategies, we required a labeled dataset for measurement and evaluation purposes. Unfortunately, we were unable to locate any publicly available datasets as previous research in this field did not make any datasets available.

To make it easier to populate the dataset, we developed a script that captures and processes the audio and keystroke data automatically. The script records an audio stream from the microphone, splits it into 2.8-second samples, and records the keystrokes and their timestamps relative to the audio sample fig. 1.

The script also automatically discards keystrokes that are too close to the splitting boundary in order to avoid having an incomplete waveform. This allows us to efficiently populate and update the dataset reducing the manual intervention during the data collection and processing.

Although the specific format of the dataset is not essential to the study, it is worth noting that we are using a lossless format to store the audio samples in our dataset, which were recorded at a sample rate of 44.1 kHz in mono.

To avoid introducing any undesired bias, such as a user's tendency to press a key in a particular way, we randomized the force and speed at which the keys were pressed while populating the dataset. This helped to ensure that the data was as unbiased as possible.

Additionally, it's worth mentioning that the samples in the dataset were sourced from a Keychron K1 mechanical keyboard, with an equal distribution of samples among classes (unique keys), the dataset is composed of 413 audio samples and 2138 keystrokes.

3.2 Finding the keystrokes

The goal of this phase is to exactly localize the keystrokes in the audio samples and isolate them. Then, in the next phase, we will further refine the isolated keystroke sounds preparing them for the computation of the similarity metrics.

3.2.1 Data Exploration

We performed keystroke detection according to a series of observations. The waveform of a keystroke presents two distinct peaks fig. 1, corresponding to the pressure and the release of the physical key. Those two peaks are respectively referred to as *press peak* and *release peak*. Similar to other works [6] [1], we decided to look for the press peak since it is generally more prominent and thus easier to detect.

Moving to the frequency domain fig. 2 we can also see that the power of a keystroke sound is spread out across all frequencies to some degree.

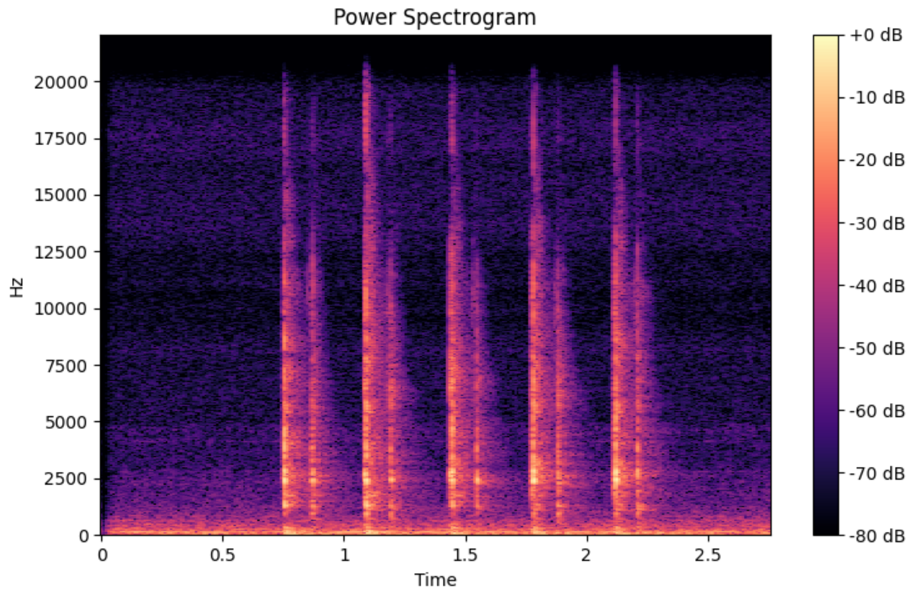

Figure 2: Power spectrum of the sample in fig. 1

3.2.2 Keystroke Detection

To identify keystrokes, we employ a method similar to that used in [6]. We use a rolling window of size 10ms, summing the FFT coefficients within the window to calculate the power of each window. This results in a 1-dimensional signal referred to as the "power signal" (S_{power}). Additionally, we compute the "sparsity signal" ($S_{sparsity}$), which indicates the distribution of signal power across all frequencies. We use a windowed mechanism as before, but instead of summing the FFT coefficients, we measure their sparsity according to eq. (1). The pseudocode for the generalized windowed operation, with a window size of $2 * w_h$, can be found in Algorithm (1). We then generate the "combination signal" S as defined in eq. (2), using two adjustable parameters k_1 and k_2.

$$S_{sparsity} = \frac{|a|_1}{|a|_2} \quad \text{with} \quad a = [a_0, a_1, ..., a_n], \tag{1}$$

$$S = S_{power}^{\circ\, k_1} \odot S_{sparsity}^{\circ\, k_2} \tag{2}$$

Algorithm 1: Windowed operation

```
Input: time domain signal $x(m)$, sampling rate $F_s$, window size $2 * w_h$
       samples
Output: resulting time domain signal $S(n)$
begin
      // Construct the real-valued matrix X discarding phase information ;
1     $X_{n,k} \leftarrow$ abs([Short-time Fourier transform of $x(m)$]) ;
      $S = [\,]$ ;
2     for $n \in [w_h, N - w_h]$ do
          // Slice X and flatten the window ;
3         $W_i \leftarrow$ flatten($X_{n-w_h:n+w_h,k}$) ;
          // Calculate value for the given window ;
4         $s \leftarrow$ operation($W_i$) ;
          // Update the signal vector ;
5         $S \leftarrow$ append($S, s$) ;
6         $n \leftarrow n + 1$ ;
      end
      // Apply Min-Max normalization to the signal ;
7     $S \leftarrow$ [(s - min(S))/(max(S) - min(S)) for s in S]
end
```

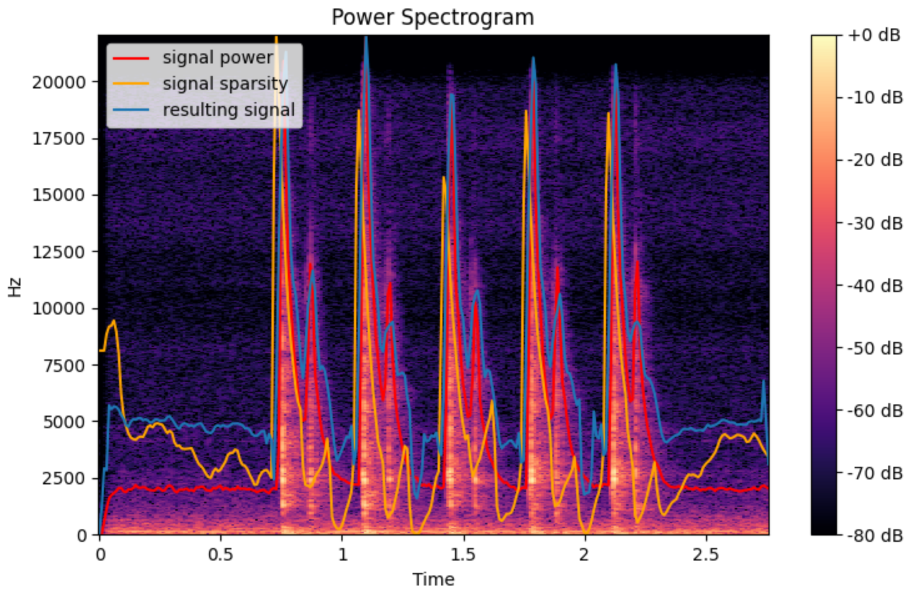

Figure 3: The "power signal", "sparsity signal" and the "resulting signal" computed for sample in fig. 1

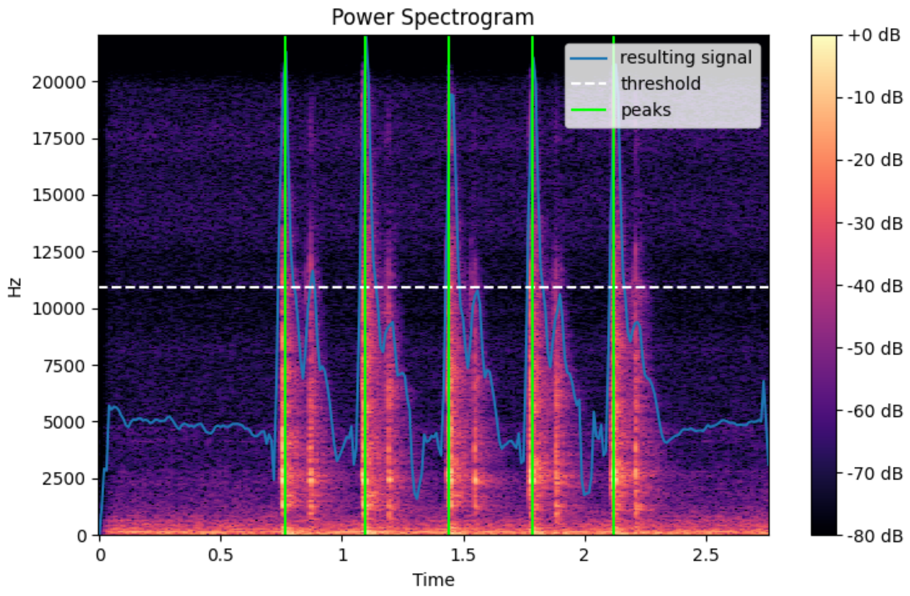

Figure 4: Keystroke detection mechanism: all four keystroke events result in having a "combination signal" (*resulting signal*) over the threshold

We then apply a threshold with tunable value T on the normalized *resulting signal* S identifying the signal peak as the maximum value enclosed in the two intersections with the threshold imaginary line fig. 4, to avoid detecting the release peak we introduce a cooldown delay every time detection occurs, we will call this tunable parameter δ.

We define the true positives as the instances where a peak is detected in the neighborhood of an actual keystroke event 60ms, the false positives as the instances where the peak is nowhere near any keystroke event, and the false negatives as the number of actual keystroke events which are further than 60ms to any peak. For sake of simplicity, we define the instance where the same peak is the closest peak for multiple keystroke events as a single true positive. Note that we cannot easily define the true negatives. We can now find our parameters k_1, k_2, and T that maximize the F1 score eq. (3), which differ from the accuracy in not requiring the true negatives.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3}$$

Where the precision is calculated as $\frac{TP}{TP+FP}$ and the recall as $\frac{TP}{TP+FN}$.

Our goal is to optimize the parameters by using Gaussian Process to maximize F1, a common method in machine learning for tuning hyperparameters.

With the set-up described above, after tuning the parameters we achieved an F1 score of 92%, and the optimal parameters were found to be $\delta = 122ms,\ k_1 = 0.79,\ k_2 = 0.21,\ T = 0.49$.

We have now developed a method for accurately identifying and isolating keystrokes. In the next phase, we will use the keystroke event locations identified in this phase as they were found to be more accurate than the timings detected by our script. We hypothesize that factors such as the GIL (python global lock for threading), setup time to connect to the recording device, and potentially an overloaded CPU may have contributed to slightly offsetting the recorded timings.

3.3 Features Extraction

In this phase, we examined the power spectrum of the signal corresponding to a keystroke event and, similarly to previous studies [6] [8], found that the most important information is located in the low frequencies, somewhere between 100Hz to 1KHz fig. 5.

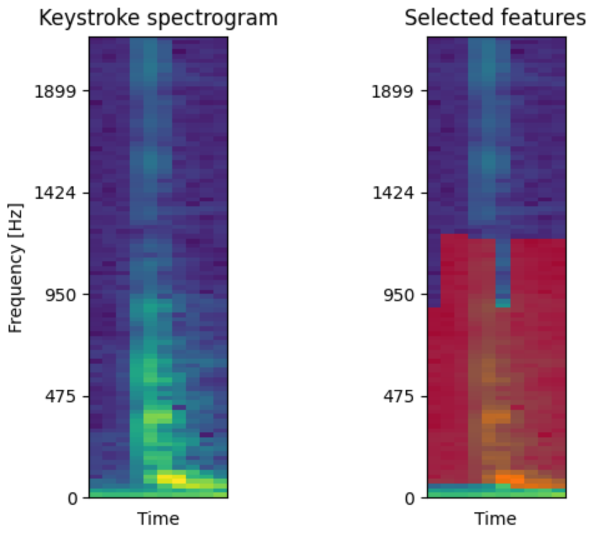

Figure 5: Selected features (in red) for an isolated keystroke power spectrum

Previous studies, such as [6], used the ”mel spectrogram,” which is a spectrogram where the frequencies are converted to the ”mel scale,” a logarithmic scale of frequencies which is believed to produce better results than a regular spectrogram. [9] Although the rationale behind this belief was not clearly explained in the cited work, we assume it produced better results because the mel scale compresses high frequencies and expands low frequencies, giving more weight to them. We prove this in fig. 6 by sequentially removing frequencies above a threshold (highcut) below (lowcut) and evaluating a logistic regression model on the labeled data using 5-fold cross-validation. We find that the linear scale spectrum performs equally well or better than the mel-spectrum. Given our observations, we conjecture that the removal of uninformative features related to high frequencies reduces overfitting, to back our argument we also found a strong correlation between the number of features not discarded by the optimal lowpass filter (highcut) and the features for the mel-spectrum fig. 7.

By using the Logistic Regression model and applying recursive elimination through 5-fold cross-validation on single features for our isolated keystrokes spectrum, we found that the selected features are very compact and focused on the lower frequencies, as expected. We also discovered that frequencies below 150 Hz are not informative and removing them improves accuracy, as shown in fig. 5.

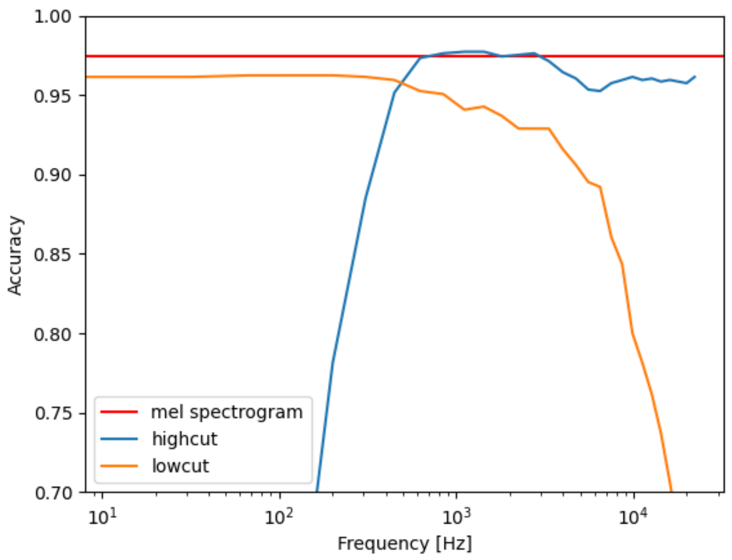

Figure 6: Performance of highcut and lowcut filters for a fixed frequency against mel spectrogram

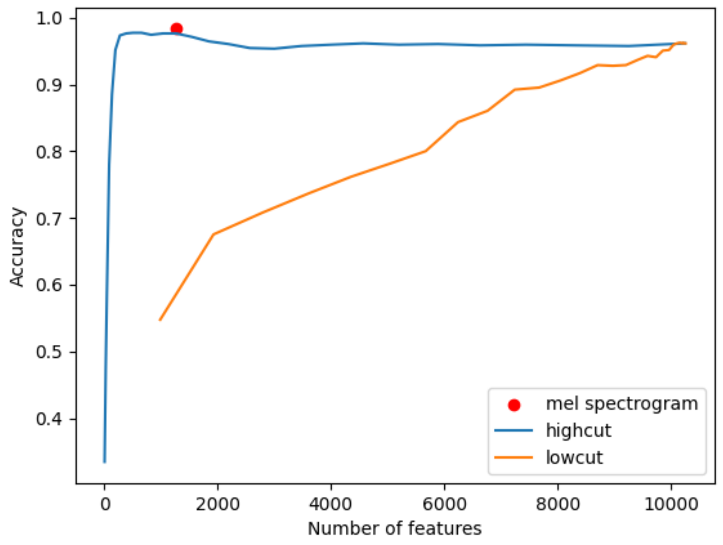

Figure 7: Performance of highcut and lowcut filters for a fixed number of features against mel spectrogram

3.4 Similarity Metrics

After extracting the features by computing the Short-Time Fourier Transform, we isolate the keystrokes and select the informative features within the range of 150-1300Hz. To compute the similarity between any two keystrokes, we evaluated a variety of metrics, including some that are specifically used for comparing the similarity between two images, as the power spectrum can be thought of as a grayscale image. To assess these metrics, we employed the Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) metric, and the results are illustrated in fig. 8. The cosine similarity metric emerges as the clear winner also having the added benefit of being very cheap in terms of computational resources.

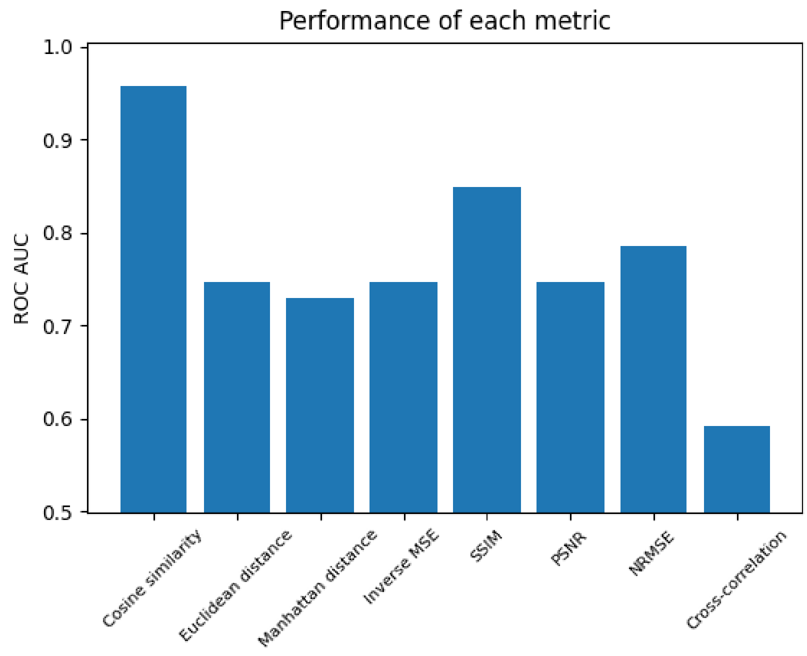

Figure 8: Performance of each metric

As seen in fig. 9, a histogram of the cosine similarity signal is presented where the optimal threshold that maximizes the F1 score is highlighted. The next step is to distinguish the two distributions without access to the true labels of the samples, in order to compute the optimal threshold. By assuming that the two distributions belong to the gamma family eq. (4), based on their shape, we can fit a Gamma Mixture Model to the combination of the two distributions in order to recreate them. Since we couldn't find any existing implementation that met our requirements, we developed our own. This also allowed us to incorporate additional constraints to improve the fitting, such as ensuring that the mode of the largest distribution is very close to the maximum value of the histogram, and that the "same label" distribution is proportionally smaller than the "different label" distribution with an increasing number of different classes in the keystrokes, and that the mode of the "same label" distribution is necessary to the right of the other. Additionally, we incorporate a regularization parameter α in the loss function at eq. (5) which prioritizes the solutions with a smaller "same label" distribution.

$$f(x;k,\theta) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}} \tag{4}$$

$$Loss(x, y, k, \delta_1, \theta_1, \delta_2, \theta_2, ratio) = MSE(x, y, k, \delta_1, \theta_1, \delta_2, \theta_2, ratio) + \alpha \cdot ratio \tag{5}$$

where:

x	= vector of bins for the histogram
y	= vector of bin sizes for the histogram, normalized from 0 to 1
k	= shape factor, equal for both distributions
δ_1	= offset for the primary distribution
θ_1	= scale factor for the secondary distribution
δ_2	= offset for the secondary distribution
θ_1	= scale factor for the secondary distribution
$ratio$	= ratio between the area of the secondary distribution to the primary
α	= regularization parameter
MSE	= the Mean Squared Error of the predicted bin sizes

additional constraints:

δ_1	= argmax(y)
δ_2	= greater than δ_1
k	= between 5 and 15

Once we specified our loss function, we were able to determine the optimal parameters. We found that utilizing the default optimization algorithm in `scikit-learn` [10], which employs the "Nelder-Mead" method [11], was sufficient for our case. The outcome can be observed in fig. 10.

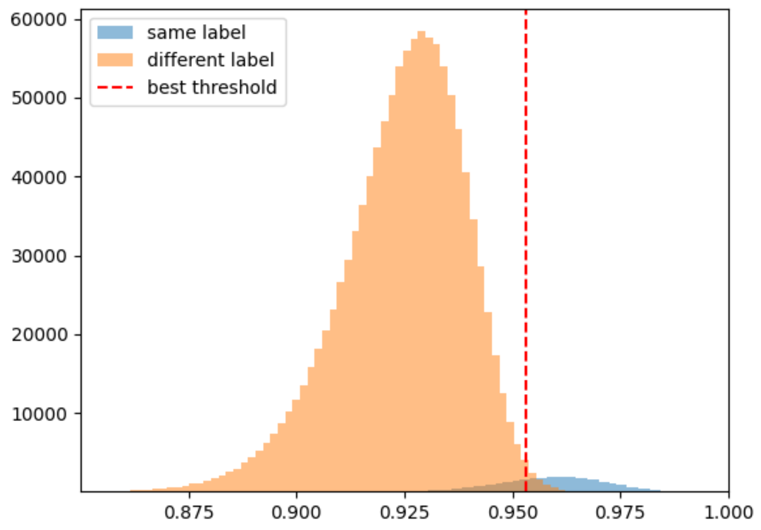

Figure 9: Histogram for cosine similarity, "same label" is the distribution of similarities where the two samples belong to the same class (or key), "different label" when they differ.

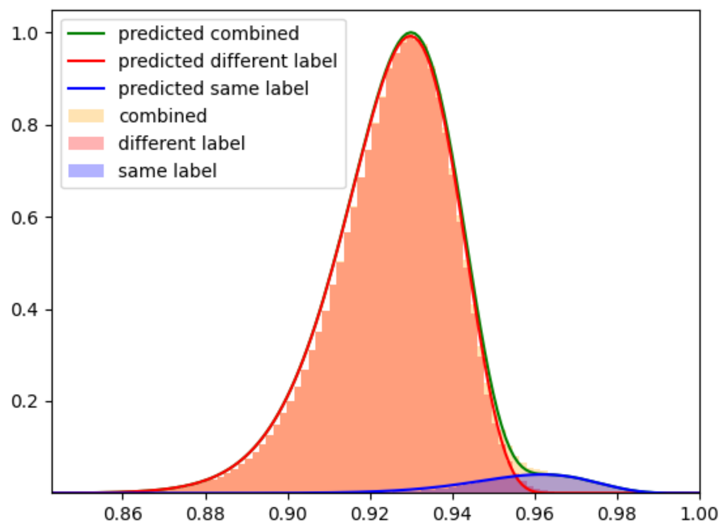

Figure 10: Reconstructed distributions from the combined distribution.

4 Evaluation

In order to evaluate the effectiveness of our methods, we created a smaller dataset from an old Asus x555 laptop that had not been used for the dataset that we used for tuning and developing our methods. Due to the reduced size of the dataset, we had to decrease the number of unique keys in order to achieve reliable results. This allowed us to test the robustness of our methods on a completely independent dataset, providing a more accurate evaluation of their effectiveness. Additionally, using a different laptop for the evaluation dataset ensured that any variations in hardware or system configurations would not impact the results of the evaluation, which was one of our primary goals.

5 Results and Analysis

The keystroke detection procedure yielded an 85% F1 score on the evaluation dataset, which is good but lower than what we expected. After investigating the issue, we discovered that the much greater variation between keystroke loudness fig. 11, probably caused by a lower quality keyboard, may have been the cause. To mitigate this problem, we propose using running statistics to normalize the signal instead of per-sample min-max normalization as a potential solution. An alternative solution would be to implement an adaptive threshold, which would adjust the threshold level for detecting keystrokes based on the current level of noise in the signal. Both of these approaches would provide a more robust solution for handling variations in keystroke loudness and would improve the overall performance of the keystroke detection system.

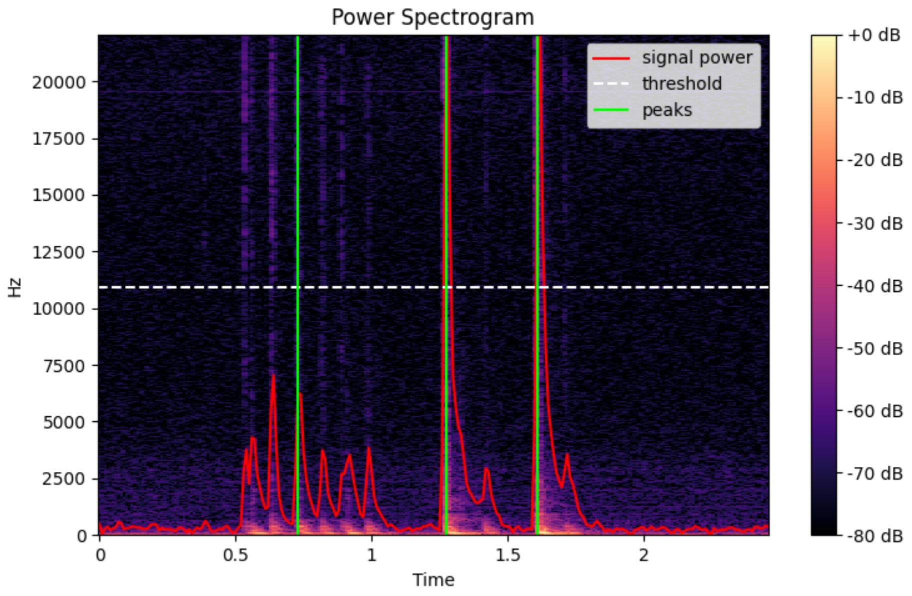

Figure 11: Evaluation dataset sample showing a wide range of keystroke loudness within a single sample

We then evaluate the similarity metric which yields similar results fig. 12 with a ROC AUC of 0.94. The threshold that maximized the F1 score was at 0.94 and the maximum F1 score was 0.72. These results were comparable to those obtained from the other dataset, which yielded a ROC AUC of 0.95, an optimal threshold of 0.95, and a maximum F1 score of 0.73.

In conclusion, the Gamma Mixture Model fitting performed well fig. 12, although we had to adjust the regularization parameter α eq. (5) to achieve the best result. This is likely due to the much larger ratio of "same label" distribution size over "different label" compared to the non-evaluation dataset fig. 9.

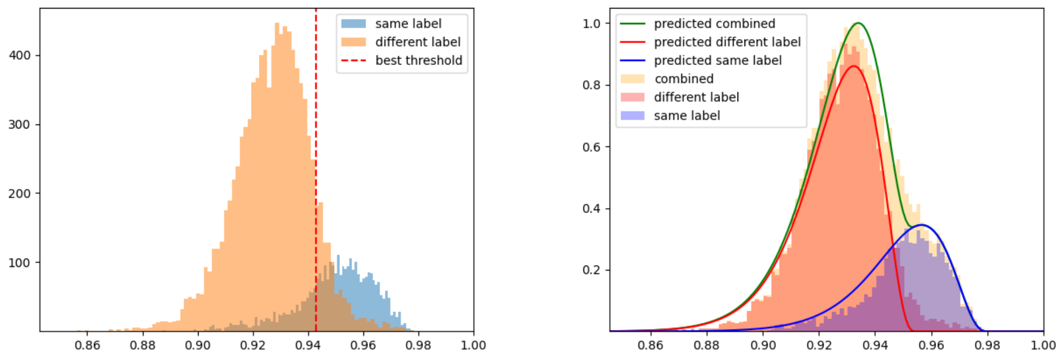

Figure 12: Histogram for cosine similarity on the evaluation dataset (left) and reconstructed histogram (right)

6 Future Work

In this section, we will explore other related areas of research that were not covered in this paper. These include potential solutions for mitigating the threat of acoustic eavesdropping and the statistical reconstruction of words based on their length and character sequence.

6.1 Possible Mitigations

We propose a solution to counter the threat of keyboard acoustic eavesdropping attacks by utilizing virtual microphones. A virtual microphone is a simulated physical microphone that can be recognized and used by most software programs. By creating a virtual microphone that preprocesses the audio signal cutting off the informative frequencies, sensitive data can be protected from malicious actors. The frequency range of human speech in a typical conversation is roughly 200 Hz to 6 kHz [12], which includes the frequencies that the filter would cut off. By only applying the filter during keystroke detection, however, the threat of acoustic eavesdropping can be eliminated while minimizing any perceived loss of audio quality.

6.2 Statistical Word Reconstruction

The ultimate goal of keystroke recognition is to reconstruct the original text that was typed. Since not every keystroke can be accurately classified, statistics can be used to reconstruct words without knowing all of their letters.

After classification, we are left with a set of tokens that represent letters of the English alphabet. However, there is no direct correspondence between tokens and letters. But, we know with a certain probability `P` whether two tokens represent the same letter or not.

To assign letters to tokens, we can use a technique called *maximum likelihood estimation.* This is a probability-based method that involves creating a matrix of size `(N. letters x N. tokens)` containing the probability of each token representing each letter, using the probability P mentioned before. Then, an optimization algorithm can be used to find the maximum likelihood of letter-to-token assignments. The assignments with the highest likelihood are then used to reconstruct the words.

It is important to note that this method is only applicable to letters of the English alphabet and that different methods must be used for recognizing other characters.

7 Discussion and Ethics

The topic of keyboard acoustic eavesdropping during conference calls has been a widely discussed issue in recent years. Various strategies have been explored, with each new work bringing new improvements. However, there have been concerns about the ethics of publicly discussing this type of attack and its potential for replication by malicious actors to steal sensitive information.

We do not believe that hiding a problem is an effective solution. We reject the idea of "security by obscurity," which is widely considered poor practice in the cybersecurity community. Our decision to study and write about this threat is to raise awareness and encourage future research, with the goal of enabling others to develop and adopt mitigation strategies to address this issue.

8 Conclusion

In this work, we designed a keyboard eavesdropping attack. We first discussed, in Section section 3.1 the setup we used to collect the data used in the evaluation of our attack model. In section 3.2 we explained in detail the mechanism used to detect keystroke events in an audio sample by incorporating the sparsity signal as an additional information source. In section 3.3 we presented our feature extraction and selection phase with many differences to most other works and explained the reasoning and the results behind our choices. In section 3.4 we evaluated different similarity metrics to correctly classify the keystrokes and found that the cosine similarity performed the best. Finally, in section 5 we evaluated our model on a separate evaluation dataset created with a distinct setup and analyzed the results, highlighting potential areas for improvement.

We believe that our work contributes to the current research in the field by discussing important aspects of a real-world applicable attack in detail. We also discussed countermeasures and provided starting points for future work, while keeping in mind the ethical aspect of our work.

References

[1] D. Asonov and R. Agrawal, “Keyboard acoustic emanations,” in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004.* IEEE. doi: 10.1109/SECPRI.2004.1301311. ISBN 978-0-7695-2136-7 pp. 3–11. [Online]. Available: http://ieeexplore.ieee.org/document/1301311/

[2] T. Halevi and N. Saxena, “A closer look at keyboard acoustic emanations: random passwords, typing styles and decoding techniques,” in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS ’12. Association for Computing Machinery. doi: 10.1145/2414456.2414509. ISBN 978-1-4503-1648-4 pp. 89–90. [Online]. Available: https://doi.org/10.1145/2414456.2414509

[3] 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 *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking.* ACM. doi: 10.1145/2789168.2790122. ISBN 978-1-4503-3619-2 pp. 142–154. [Online]. Available: https://dl.acm.org/doi/10.1145/2789168.2790122

[4] Z. Martinasek, V. Clupek, and K. Trasy, “Acoustic attack on keyboard using spectrogram and neural network,” in *2015 38th International Conference on Telecommunications and Signal Processing (TSP).* doi: 10.1109/TSP.2015.7296341 pp. 637–641.

[5] L. Zhuang, F. Zhou, and J. D. Tygar, “Keyboard acoustic emanations revisited,” in *ACM Transactions on Information and System Security*, vol. 13. doi: 10.1145/1609956.1609959 pp. 1–26. [Online]. Available: https://doi.org/10.1145/1609956.1609959

[6] A. Compagno, M. Conti, D. Lain, and G. Tsudik, “Don’t skype & type!: Acoustic eavesdropping in voice-over-IP,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security.* ACM. doi: 10.1145/3052973.3053005. ISBN 978-1-4503-4944-4 pp. 703–715. [Online]. Available: https://dl.acm.org/doi/10.1145/3052973.3053005

[7] Y. Berger, A. Wool, and A. Yeredor, “Dictionary attacks using keyboard acoustic emanations,” in *Proceedings of the 13th ACM conference on Computer and communications security - CCS ’06.* ACM Press. doi: 10.1145/1180405.1180436. ISBN 978-1-59593-518-2 pp. 245–254. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1180405.1180436

[8] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM. doi: 10.1145/2660267.2660296. ISBN 978-1-4503-2957-6 pp. 453–464. [Online]. Available: https://dl.acm.org/doi/10.1145/2660267.2660296

[9] N. Kamarudin, S. Al-Haddad, S. J. Hashim, M. A. Nematollahi, and A. R. Bin Hassan, "Feature extraction using spectral centroid and mel frequency cepstral coefficient for quranic accent automatic identification," in *2014 IEEE Student Conference on Research and Development*. doi: 10.1109/SCORED.2014.7072945 pp. 1–6.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in {P}ython," vol. 12, pp. 2825–2830.

[11] J. A. Nelder and R. Mead, "A simplex method for function minimization," vol. 7, no. 4, pp. 308–313. doi: 10.1093/comjnl/7.4.308. [Online]. Available: https://doi.org/10.1093/comjnl/7.4.308

[12] R. Quam, I. Martínez, C. Lorenzo, B. A, M. Rosa-Zurera, J. P, and A. JL, "Studying audition in fossil hominins: A new approach to the evolution of language?" in *Psychology of Language*. ISBN 978-1-61942-819-5 pp. 1–37, journal Abbreviation: Psychology of Language.