Report: Multi-Modal Side-Channel Attack on Keyboard Presses

Yichen Cai, *University of Toronto*

Abstract

Side-channel attacks exploit the information leaked by normal user activities to infer sensitive data. It poses a threat to cybersecurity as it often leaks data in unexpected ways. This project presents a novel approach to leaking keystroke information using a multi-modal machine learning algorithm that processes both audio and video inputs. The audio input is segmented by an improved algorithm that detects and estimates the time window for each recorded keystroke. And with the time window, we can easily predict the exact frame of the keypress within the video input. Both inputs are processed with their respective neural networks and their results are weighted by an attention network which balances the two modalities. Finally, the entire detected string is refined by a pre-trained GPT model so that it enhances coherence and denoises the string. This integrated approach demonstrates significant potential in improving the accuracy of keystroke side-channel attacks. Project code is located at: https://github.com/cycv5/SideChannelOnKeyboard

1. Introduction

Side-channel attacks have often emerged as unexpected ways to leak sensitive information in cybersecurity. These attacks leverage the unintended channels of emission like electromagnetic radiation, power consumption, or acoustic data, to infer the actual data being processed. Acoustic side-channel attacks have been explored and gained significant attention as it can capture keystroke sounds and deduce the corresponding key.

This paper explores a novel approach to keystroke side-channel inference by utilizing a multi-modal machine learning algorithm that analyzes both acoustic and visual information. Since video and audio capturing devices have become easily accessible, it is feasible to get both video and audio data from a person typing on a keyboard. This can be done through a webcam (as in this project), a smartphone, or a specially made hidden camera/microphone device. The data are then preprocessed by a custom algorithm to extract the audio and video frame of the key presses. And the two types of input are processed independently first and combined later.

To address the potential noise in the inference due to misclassification, a pre-train large language model (LLM) is used (GPT 3.5) to refine, correct and clean the output. This enhances the overall coherence of the output data. The integrated method gives higher accuracy and more robustness because of the LLM.

2. Related Work

Previous works on this topic have focused on a single channel of attack.

A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards [1]

This paper proposed a method of extracting the data from keyboard strokes audio. They recorded audio of people typing on a keyboard and partitioned into fixed length keystrokes. Then those data are augmented and trained with a CoAtNet - a combination of CNN and the attention mechanism. They found good accuracy with this method (95% accuracy). However, this method requires clear recording of the typing sound and errors are seen for keys that are close to each other.

Towards a General Video-based Keystroke Inference Attack [2]

This paper proposed a video-based method for detecting keystrokes. It uses hand recognition to build a model of touch points on a plane and determine which keys are pressed. The paper measures the semantic similarity and shows that with an indoor setting, the similarity score can be 98%+. However, this requires a clear detection of key presses (and a frontal view of the typer) - without additional visual distractions.

3. Method

The combination of the audio and video input can create a more robust yet simpler detection algorithm. The audio data can help pinpoint the exact time of key presses, thus

giving us the timing information for a keypress. This, in a way, denoises the video feed and aids us in finding the corresponding frames in the video. The video, at the same time, gives a direct view of the hand positions that helps with the final classification result, paired with the audio data. This is especially helpful to distinguish key presses that are close to each other. Finally, a pre-trained LLM model will be employed to fix any noise in the output.

3.1. Data Preprocess

Audio and video data captured must be preprocessed for the downstream tasks to be performed on them. The two forms of data are first separated.

Audio data is first transformed by Short-Time Fourier Transform (STFT), which is then used to calculate the Root Mean Square (RMS) energy for identifying the moments of key presses [1]. Figure 1 demonstrates the RMS energy graph for 30 key presses.

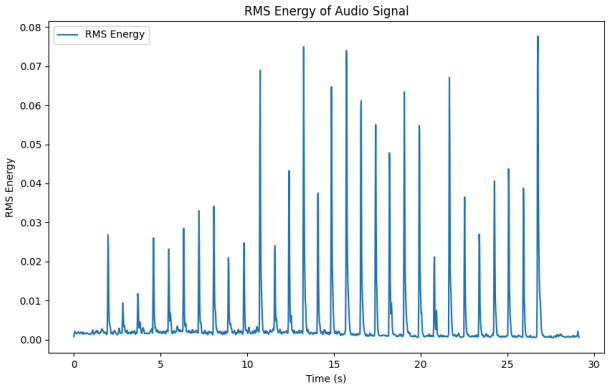

Figure 1: RMS energy of 30 key presses.

Previous methods used a rough fixed estimation of the key press time window (around 0.3 seconds). In this project, a new algorithm is developed for variable length keypress detection. Given the RMS energy graph, remove the background noise by ignoring anything lower than a certain threshold energy level (e.g., 0.002). Then iterate through the energy graph and compute the rough windows of key presses by identifying where the energy level drops and rises again. In other words, by identifying the gaps between key presses, we know the "inverse" is the keypress windows. Make sure that each keypress window has a maximum energy level that surpasses another higher threshold (e.g., 0.005) so that it will filter out noisy windows where no keypress happened. The rough time windows that we have now will have some cut off at the beginning and the end due to the thresholding. The algorithm then tries to expand the two ends of the window if the energy level keeps falling, i.e., no new peaks happen, meaning that the sound produced is a part of the same key press. Now we have segmented the entire audio data into different keypresses. Mel-spectrogram is used to visualize the key strokes as it can capture the fine details in the audio data.

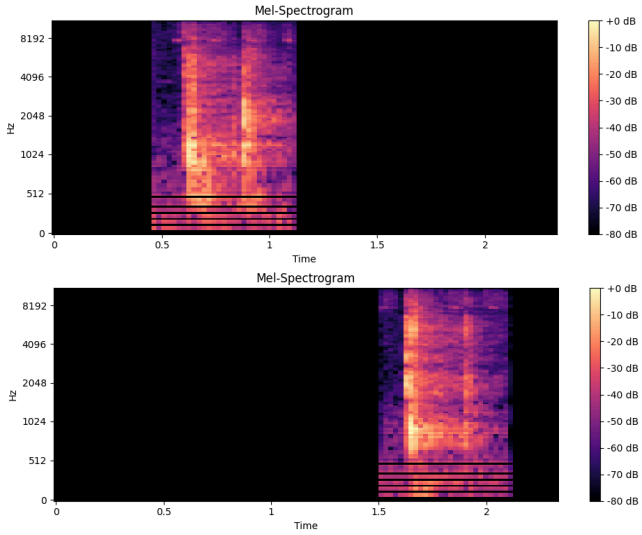

Figure 2: Mel-spectrogram of typing letter A and P. Black areas are padding.

With the identified time window for each keypress, we identified through experimentation that the exact video frame of the keypress is right before the middle of the window. The video frame is extracted and a homography is calculated to correct the perspective of the keyboard. This is to ensure that we always have a rectangular representation of the keyboard and the viewing angle is consistent no matter what angle the original video has. Figure 3 shows the perspective-corrected image.

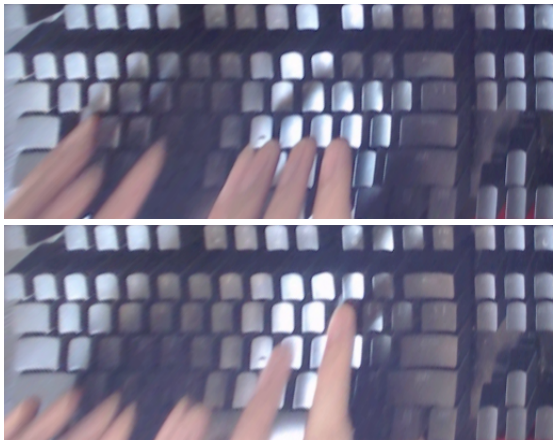

Figure 3: Perspective Corrected video frames of typing the letter A and P.

With the Mel-spectrogram of the audio and the corrected video frames, the data is ready for processing by the neural network.

3.2. Neural Network

The Mel-spectrogram of the audio is then input to a custom CNN network with 2 convolution layers (with max pooling and ReLU activations) and 2 fully connected layers (with dropouts). The output is a 128 dimension latent embedding of the audio information.

The video input is passed through a ResNet-18 and also outputs a 128 dimension embedding for further processing [3].

3.3. Attention Mechanism

Since the two embeddings (from audio and video frame) will jointly help make the final decision, an attention mechanism is used to determine the weight to put on each embedding when making the final decision. The attention is made up of two fully connected layers outputting 2 normalized attention weights for the audio and video frame embedding. The weights determine how much emphasis will be put on each form of input, and it is learned during the training so that different keypresses would have different weights. This way, every keypress can rely on the more differentiating piece of information to determine the output while the other input can help rule out incorrect classes when needed.

3.4. Training Coordination for Multi-modal Model

When training a multi-modal model, it is often found that the model relies on only one form of input while ignoring the other. And this is found during the training phase of this project as well. The model initially puts attention (almost) solely on the video frame embeddings. And that is expected due to the fact that the video frame contains much more information than the sound - and it is easy to make predictions on the keypresses just based on the image. Therefore, the video network quickly gains "traction" while the audio pipeline still struggles to produce meaningful inferences. However, by testing only the audio pipeline, we can observe that the audio information does provide meaningful outputs (i.e., decrease in loss and increase in training/validation accuracies) - it just requires more training than the video frame.

In order to synchronize the training and effectively utilize both forms of input, 2 measures are put in place:

1. The audio model is pre-trained by 100 epochs to make up for the fact that it needs more time to train than the video pipeline. Then the combined input will be passed into the model to be trained.

2. A regularization step is added to the loss, so that if the attention weights are largely skewed towards one form of input, the model will be penalized by a larger loss. The regularization loss is the squared difference between the 2 attention weights, multiplied by a tunable hyperparameter λ (e.g., $\lambda = 0.04$).

3.5. GPT Integration for Coherence

The output string of a series of keypresses can be noisy if the input leads to some misclassifications. It also lacks punctuation and capitalization. Instead of increasing the model complexity, pre-trained LLMs like GPT-3.5 by OpenAI can effectively correct those noises and make accurate predictions in punctuations and capitalizations. The LLM is trained on large text corpus containing natural human languages. Given a noisy text, it can recover the original sentence with extremely high accuracy.

4. Results

4.1. Training Results

The training is split into 2 stages as described in Section 3.4. The first part of the training is done only on the audio input.

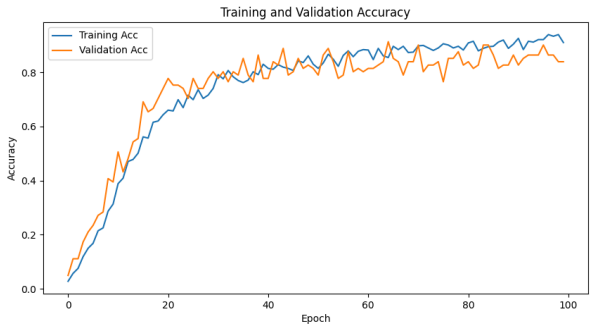

Figure 4: Training and validation accuracy for only audio input.

From Figure 4, it is clear that the audio input does contain critical information that helps the classification task. The training accuracy can reach 90%+ and the validation accuracy converges to around 85%.

Then the combined training is conducted for the final result.

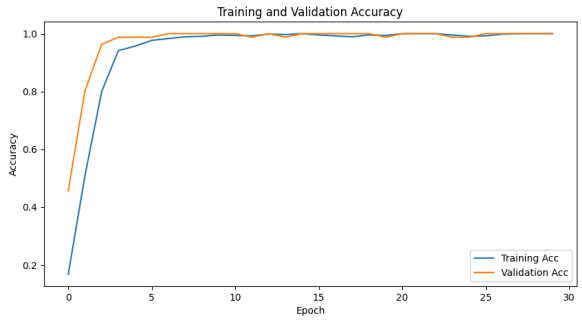

Figure 5: Training and validation accuracy for the combined training stage.

With pre-trained weights from the audio training, the combined training converges much quicker. After around 10 epochs, it already reaches 99%+ training and validation accuracy. This shows that the model is not overfitting, instead, it learns the boundaries between different key presses and can classify them very well.

4.2. Testing Results

For testing, another 81 new, randomly selected key presses are chosen. The trained model classifies those 81 key presses with 100% accuracy. The attention weights put on the two forms of input have very different values, which shows that different key presses put emphasis on different sources of input. Furthermore, they do not take on extreme values (e.g., values very close to 1 or 0) shows that they utilize both inputs to make the final decision.

However, it is worth noting that test audio and video are taken under similar settings as the training and on the same keyboard. But noticeable differences can be observed in terms of ambient lighting and keypress pressures (as keypresses made by hands are inconsistent and the sounds made by the presses are inherently random). Therefore, the test results are meaningful.

4.3. Power of Multiple Modality

To see the power of multi-modality, we analyze the validation performance of individual forms of input and the combined. As shown in Figure 6, the accuracy on the validation set (a set that the trained model does not see, but we tuned the parameters on) is the highest for the combined model. This shows that the 2 modality combined provided better results compared to a single input.

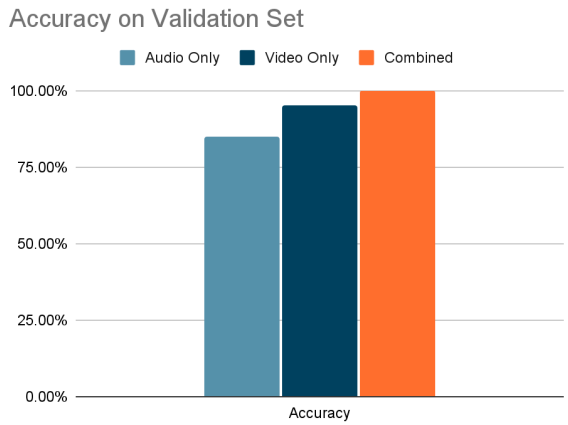

Figure 6: Validation accuracy with different sources of input.

4.4. Organization by GPT

The GPT3.5 by OpenAI corrects small mistakes and organizes the output. Although the test set reaches 100% accuracy, we did discover some misclassification during the training and validation, which could lead to errors in future testing.

For example, a common error is misclassification between class 0 and 18 (letter A and S respectively). The two letters are close to each other in the middle row (of a Qwerty keyboard), hence confusions are likely. A test sentence *"i like apples and bananas"* can be recovered, with some probability, as *"i like spples and bananss"*. We put the recovered sentence into GPT 3.5 with the following prompt: *"Please check for errors and respond with a correct and clean version of the following (and only the following)"*. The GPT returns "*I like apples and bananas."*

GPT also provides proper capitalization and punctuations. For example, a recovered sentence is *"to get to the parkm i took a busl and after that i had some apples bananas and orangesm"*. Notice that there are no capitalization and the punctuations are incorrectly classified as some letters (l and m). The corrected sentence by GPT is: *"To get to the park, I took a bus, and after that, I had some apples, bananas, and oranges."*

It is interesting that human typing errors often coincide with misclassifications. Human errors usually involve pressing the wrong key next to the correct one. This sort of behavior is expected by the GPT algorithm. The misclassifications of our model demonstrate the exact

same behavior as the keys that are close to each other are the hardest to tell apart.

5. Discussion

5.1. Discussion on the Result

The results of the project shows that the multi-modal method of keypress classification is highly effective. It gives a perfect testing accuracy. And it also shows that it surpasses the capability of a single source algorithm. This means that with a hidden camera and microphone, an attacker can gain secret information by monitoring the keypresses on a keyboard. This poses a severe security threat that people should be aware of. It is also worth noting that due to the simplicity of the model (and advancement in computing power), the training of such a model can be done in under 10 minutes (around 8 minutes in our setting). This means that even if we train the model from scratch, the system can be deployed in a relatively short time period.

5.2. Limitations

There are also some limitations on this model. For example, the relative position between the keyboard and the camera is roughly the same for our dataset. A drastically different setting would have an impact on the classification accuracy. However, with a more extensive dataset trained on a specific target, it is highly likely that the model would adapt to different settings - just like any other state-of-the-art classification algorithm. Even with this limitation, there are still many real world applications where this model would be sufficient. For example, training on the keypad of an ATM machine where the keypads are fixed in place and a hidden camera can be placed.

5.3. Countermeasures

Countermeasures of this attack can be divided into two parts:

Since ML models are data-hungry, stopping data collection prevents such models from being trained. This can be scanning for hidden cameras, or designing specific communication methods that provide very little side-channel information.

Another aspect is to stop the model from getting data for inference. For example, use your hand to block any vision when entering the password in public, or turning off the microphone and camera when you are entering password on a video call/live stream.

In general, the awareness that you are potential on camera can help mitigate the negative effect of such attacks.

6. Conclusion

In conclusion, this project presents a new approach to keystroke inference through the integration of multi-modal machine learning techniques, using both audio and video data.

By capturing the acoustic and visual signals of a person typing, the data is first preprocessed with a custom algorithm. Then the audio model is trained first. The combined training follows and an attention mechanism with regularization is used to ensure a balanced and accurate calculation. To mitigate the noise and potential misclassifications, a Generative Pre-trained Transformer (GPT) model is employed to improve the coherence of the output text.

The findings show that the proposed model gives a great result in inferring the text typed by the victim. And the entire model can be trained and deployed within minutes. It serves as a reminder that it is critical to put in place countermeasures to ensure the safety of our sensitive data.

7. References

[1] J. Harrison, E. Toreini, and M. Mehrnezhad, “A practical deep learning-based acoustic side channel attack on keyboards,” 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), vol. 7, pp. 270–280, Jul. 2023. doi:10.1109/eurospw59978.2023.00034

[2] Z. Yang, Y. Chen, Z. Sarwar, H. Schwartz, B. Y. Zhao, and H. Zheng, ‘Towards a General Video-based Keystroke Inference Attack’, in 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 141–158.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016. doi:10.1109/cvpr.2016.90

8. Appendix

8.1. Data Collection Methods

Data collection (recording) is done with a Logitech C922 webcam (for audio and video). And typing is done on a Razer optical switch keyboard (no additional clicking aside from the switches and the impact of the keycaps with the frame).

The camera is placed so that it can capture a (side-) frontal view of the typer's hand. Each recording contains 30 keystrokes from the typer. In total, 1620 individual keypresses are recorded for 26 letters and the space bar.

8.2. Setup

Training, validation and testing is done on a RTX 4070 Super GPU in Windows 10 environment. The three datasets are randomly drawn from the collected data.