IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.Doi Number

A New Pipeline for Snooping Keystroke Based on Deep Learning Algorithm

Massimo Orazio Spata, Valerio Maria Russo, Alessandro Ortis, (Senior Member, IEEE), Sebastiano Battiato, (Senior Member, IEEE)

Dipartimento di Matematica e Informatica, University of Catania, 95125 Catania, Italy

Corresponding author: Massimo Orazio Spata (massimo.spata@unict.it).

ABSTRACT This research focuses on the vulnerability issues related to keystroke logging on a physical computer keyboard, known as Snooping Keystrokes. This category of attacks occurs recording an audio track with a smartphone while typing on the keyboard, and processing the audio to detect individual pressed keys. To address this issue, mathematical wavelet transforms have been tested, and key recognition has been implemented using the inference test of a deep learning model based on a Temporal Convolutional Network (TCN). The novelty of the proposed pipeline lies in its dynamic audio analysis and keystroke recognition, which splits the wave based on audio signal peaks generated by key presses. This approach enables an attack in real-world conditions without knowing the exact number of keystrokes typed by the user. Experimental results for the proposed pipeline show a peak accuracy of 98.3%.

INDEX TERMS Acoustic side channel attack, Snooping keystroke attacks, Deep learning, User security and privacy, Laptop keystroke attacks, Zoom-based acoustic attacks

I. INTRODUCTION

The increasing use of smart devices in our daily lives and their growing importance in managing our sensitive personal data, have heightened attention to cybersecurity. Among various threats, Acoustic Side Channel Attacks (ASCAs) have emerged as a significant concern. These attacks exploit sound emissions from devices to extract sensitive information, such as pressed keys or conversations, compromising digital security and user privacy. This research focuses on a specific type of acoustic attack known as Snooping Keystrokes. This attack aims to retrieve information about pressed keys on a nearby computer's physical keyboard, thereby extracting sensitive information like private passwords for digital devices or accounts.

In this study, the most relevant Snooping Keystrokes attacks have been analyzed in detail, including detecting individual clicks (using wavelet transform conversion of the audio track), segmenting the audio of pressed keys, extracting individual sound features representing the clicks, and recognizing the pressed keys using artificial intelligence techniques. Specifically, a Temporal Convolutional Network (TCN) [1, 19, 23, 31] model has been applied for classification. Through a series of experiments conducted on a comprehensive dataset, including various types of smart devices and attack scenarios, the performance of the detection and recognition model has been evaluated. The results demonstrate the effectiveness of the proposed method, achieving high precision, sensitivity, and specificity. Furthermore, the method's robustness and versatility in different real-world conditions are highlighted. The pipeline for acoustic attack on keyboards, is supported by a comparative evaluation with Harrison's paper [2] and by exploiting new challenges on a collected dataset. The TCN facilitates the modeling of complex temporal dependencies, enabling extraction of latent patterns within the acoustic emissions during typing.

The remainder of this paper is organized as follows: Section 2 provides a summary of previous research works related to the topic, the proposed pipeline and the details of the developed deep learning model architecture in Section 3 and the description of experimental result is presented in Section 4. Section 5 presents the conclusions based on the results, which confirmed the highly promising performance of the designed solution.

II. RELATED WORKS

Traditional methodologies for keystroke acoustic attacks [8, 9] have often relied on simplistic analysis techniques, limited in their capacity to capture complex temporal dynamics inherent in typing sounds. The addressed task, received a significant increasing attention in the scientific literature of the last years as for example works [4, 23, 26]. The oldest paper on emanation-based side-channel attacks identified in this review was authored for the United States National Security Agency (NSA) in 1972 [8]. As discussed by authors in study [9], examines a more recent keyboard featuring a slightly recessed design. Despite this, the keycaps are still large and made of plastic, significantly differing from modern laptop keyboards. The authors acknowledge that testing laptop keyboards, might yield different results due to the absence of a 'release peak' in the waveform. In the Asonov et al. study [4] mentions that classified documents generated under the NSA's side channel specification program (TEMPEST), are known to address acoustic emanations. Such work, often referred as the initial ASCA targeting a keyboard, was published in 2004. It criticized prominent plastic keyboards of that era. Despite its early position in the field, the paper successfully demon-started attacks on an ATM keypad, a corded telephone, and two keys from a laptop keyboard. With the latest advancements in deep learning models, the possibility of an acoustic attack on keyboards seems increasingly plausible, as highlighted in recent studies [6] where authors, has examined the potential for Acoustic Side-Channel Attacks (ASCAs) on laptop keyboards, especially given that laptops are a prime target for such attacks. Laptops, being more portable than desktop computers, are often used in public places where keyboard sounds can be overheard, such as libraries, coffee shops, and study areas.

In the research [23] authors discovered that several methods outperformed neural networks during testing, and attaining a keystroke accuracy of 74.3%, a result that was similarly reflected in [24]. In the paper [26] authors examined the feasibility and mechanics of acoustic side-channel attacks on keyboards, which arise from the sounds and vibrations created while typing. Authors investigated the unique sounds produced by different keys, considering the physics of keyboards and the various typing styles of users and emphasizing the practicality of these attack scenarios, showing that attackers can use both physical proximity and remote indirect methods, to record keystroke sounds. This study represents a significant threat to user security and privacy, as attackers can capture sensitive information without needing direct physical access to the victim's device.

The NSA document NACSIM 5000 [11], which was partially declassified, identified acoustic emanations as a potential security risk, in 1982. In the public domain, Acoustic Side Channel Attacks (ASCAs) have achieved different levels of success on contemporary keyboards, utilizing a wide range of techniques. In work [12] results indicating that the potential for a real world ASCA is the tendency of each classifier to group false classifications near the correct key. This characteristic, suggests that even incorrect classifications may provide clues about the true key's location on the keyboard, a feature that could be leveraged in future research. In reference to [13] author describe how, since 1950s, British spies utilizing acoustic emanation sound of the Hagelin encryption devices within the Egyptian embassy. As presented in the paper [14], the authors suggest two-factor authentication as an effective defense that has stood the test of time. This method involves using an additional device or biometric verification to access data. With the increasing inclusion of biometric scanners in laptops, the need to enter passwords via keyboard is significantly reduced, thereby diminishing the threat posed by ASCAs. Nonetheless, the risk persists that data other than passwords could still be obtained through ASCA.

Recently, a deep learning model have been used in order to classify laptop keystrokes, just using a standard smartphone integrated microphone [2]. Experiments over multiple evaluation settings shown as related overall performances outperforms a significant pool of previous works [10, 22, 24]. The model presented in the work [2] has been trained on two different datasets [21] created with keystrokes recorded by a nearby phone and the video-conferencing software Zoom, whereas classifier achieved respectively a peak accuracy of 95% and 93%. The authors exploited the CoAtNet model, a recent deep neural architecture based on attention mechanism [3].

This paper presents a novel pipeline for acoustic attack on keyboards, supported by a comparative evaluation with Harrison's study [2] and with our own other specially collected dataset. The novelty is due to the entire pipeline used and to exploitation of TCN models [1], usually applied on different tasks, for acoustic keyboard attack. The TCN facilitates the modeling of complex temporal dependencies, enabling extraction of latent patterns within the acoustic emissions during typing. Its core strength lies in the capability to capture and process sequential data, dynamically adapting to the variations in typing speed, rhythm, and inter-key intervals. This is a first step of larger research comprising benchmarking of several architectures and models. Other related approaches [6, 10, 23, 24] make use of different settings and methods, obtaining overall accuracy on different ranges. Addressing this challenge, this work focuses on a pipeline methodology for real world conditions scenarios of Keystroke Attacks.

TCN offer significant advancements over existing methods in keystroke attack detection by addressing key challenges related to temporal dependencies and overfitting. Traditional models like RNN and LSTM, while powerful, often struggle with long-range dependencies and are prone to issues like vanishing gradients. TCN mitigate these problems with their architecture, which includes causal and dilated convolutions, allowing the model to efficiently capture long-term dependencies in keystroke data without the risk of information leakage from future inputs to past outputs. One of the key

strengths of TCN is their ability to handle sequences of varying lengths, which is crucial for accurately modeling keystroke dynamics. Unlike RNN that process inputs sequentially, TCN can process entire sequences in parallel, significantly speeding up training and inference. This parallelism also reduces the risk of overfitting by providing a more comprehensive understanding of the temporal structure within the data. Overfitting is a common challenge in deep learning, where a model performs well on training data but poorly on unseen data due to excessive complexity. TCN address this through their use of residual connections and dilated convolutions. Residual connections help maintain gradient flow during backpropagation, preventing the vanishing gradient problem and allowing for deeper networks. Dilated convolutions expand the receptive field exponentially without increasing the number of parameters, enabling TCN to capture patterns over longer time spans without overfitting. Furthermore, TCN are inherently designed to handle the irregularities and variability present in keystroke dynamics. By using a combination of causal and dilated convolutions, TCN can model both short-term and long-term dependencies more effectively than traditional RNN-based models. This capability is particularly useful in detecting keystroke patterns that may vary significantly between different typing styles and contexts.

III. PROPOSED METHOD

The proposed work exploits a TCN [1] methodology as a promising approach to counter such keystroke acoustic vulnerabilities.

TCN is a convolutional neural network designed specifically for sequential data. They have gained popularity in tasks involving time series, such as audio processing, natural language processing, and any other sequence modeling problems. Key concepts of TCN are: causal convolutions, dilated convolution and sequence length flexibility.

In a TCN, convolutions are causal, meaning the output at time t only depends on the inputs from time t and earlier. This ensures that the model respects the temporal order of the data, preventing information leakage from future to past. A TCN use dilated convolutions to allow the network to have a larger receptive field without increasing the number of parameters or the computational complexity significantly. Dilations introduce gaps between the filter taps, enabling the model to capture long-range dependencies efficiently.

For these reasons, the TCN paradigm offers a groundbreaking solution by harnessing the power of deep learning and temporal convolutional architectures. This approach facilitates the modeling of complex temporal dependencies, enabling the extraction of latent patterns within the acoustic emissions during typing. The TCN's core strength lies in its capability to capture and process sequential data, as it can dynamically adapt to the variations in typing speed, rhythm, and inter-key intervals. By incorporating dilated convolutions, the TCN model can exponentially expand its receptive field, effectively integrating information from a wide temporal range. TCN not only facilitates accurate feature extraction from raw acoustic signals but also enhances the model's resilience against noise and variability. In order to provide a fair experimental comparison with respect to the state of the art, we employed the same dataset as in [2, 21, 29], as well as the same evaluation metrics. Experiments suggest that the TCN paradigm represents a promising avenue for advancing the field of cybersecurity against unconventional threats. Given a specific keyboard, the first step involves the creation of an audio dataset. Then, the entire attack system is composed by two phases: first data are properly processed in order to apply next a TCN training process on specific extracted features. During the initial experimentation phases, efforts were made to generalize the system as much as possible to demonstrate experimentally that even when imposing the worst-case scenario, excellent performance could still be achieved.

A. PRE-PROCESSING PHASE

This phase includes peak detection and splitting algorithm, followed by data augmentation. The visualization of the waves occurs using an Amplitude-Time graph. Below are two examples to visually understand how audio tracks appear. In Figure 1 and Figure 2, there has been observed several characteristics that will pose challenges to overcome during this discussion for the correct classification of click patterns. The peaks have very different values because obviously the clicks have non-uniform energy applied in the action of key pressing, and the structure of the click consisting in two peaks: the first, higher one is the actual click, and the second, generally lower, is the sound of releasing the key when we remove our finger. Analyzing the time between clicks, it is not homogeneous in terms of frequency; in a real case, keys are pressed at different frequencies from each other. Finally, the background noise is not always the same during audio recording and it is not negligible, especially in real cases of widespread ambient noise. A high level of background noise could compromise the attack. In Figure 3 the pseudo-code of peak detection and splitting algorithms: the Algorithm 1 named “Peak Detection and Split” used for peak detection and split algorithm, get in input the path of recorded audio file, the type of wavelet transforms to apply and an output path were to write splitted audio files returned by the algorithm and call recursively the Algorithm 2 named “Split Audio” which get in input the path, the recorded audio file the average peak value, the average inter-click time during typing and the output file for each peak detected.

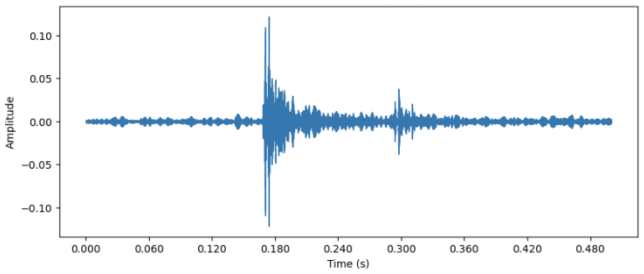

FIGURE 1. **An audio wave with a single click recorded by a smartphone during experiments.**

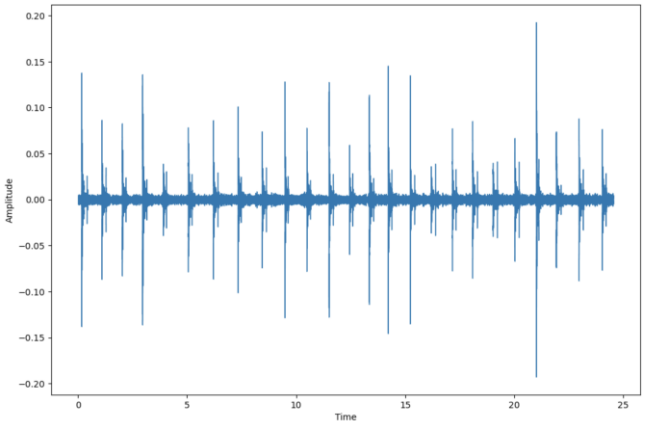

FIGURE 2. **An audio wave track with 25 clicks on the keyboard recorded by smartphone during experiments.**

Algorithm 1 Peak Detection and Split Algorithm

```
function PEAK_DETECTION_AND_SPLIT(file_path, waveletname, output_dir)
    click_time ← 0.3
    y, sr ← librs.load(file_path)
    coefficients, frequencies ← pywt.cwt(y, np.arange(1, 128), waveletname)
    noise_level ← np.mean(np.abs(coefficients[:, : int(sr)]))
    threshold ← 5 × noise_level
    i ← 0
    peak_times ← []
    peak_count ← 0
    while i < len(coefficients[0]) do
        for j in range(len(coefficients)) do
            maybe_peak ← abs(coefficients[j, i] − coefficients[j, i − 1])
            if i > 0 and maybe_peak > threshold then
                time_of_peak ← librs.samples_to_time(i, sr = sr)
                peak_times.append(time_of_peak)
                i ← i + int(click_time × sr)
                peak_count ← peak_count + 1
                break
            end if
        end for
        i ← i + 1
    end while
    split_audio(file_path, peak_times, click_time, output_dir)
end function
```

Algorithm 2 Split Audio Algorithm

```
function SPLIT_AUDIO(file_path, peak_times, click_time, output_dir)
    audio ← AudioSegment.from_file(file_path)
    t ← 0
    while t < len(peak_times) do
        start_time ← int(peak_times[t] × 1000)
        if start_time < 0 then
            start_time ← 0
        end if
        end_time ← int((peak_times[t] + (click_time − 0.1)) × 1000)
        split_audio ← audio[start_time : end_time]
        file_name_without_extension ← os.path.splitext(os.path.basename(file_path))[0]
        output_filename ← os.path.join(output_dir, f"file_name_without_extension_split_t+1.wav")
        split_audio.export(output_filename, format = "wav")
        t ← t + 1
    end while
end function
```

FIGURE 3. **The peak detection (top) and split algorithm (bottom) pseudocodes.**

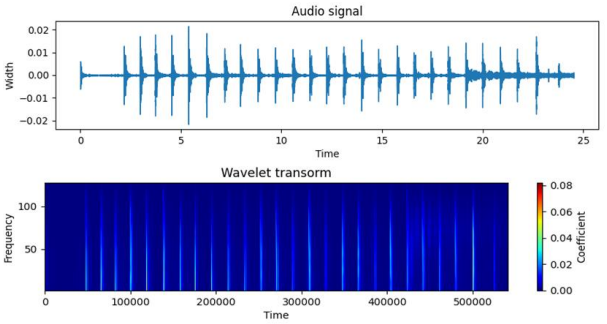

FIGURE 4. **A wavelet transforms of an audio wave sample with 25 clicks.**

IV. WAVELET TRANSFORM AND THE NEW PIPELINE

Mathematically, a wavelet [27, 28] must also satisfy the condition of admissibility in (1):

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \quad (1)$$

This means that the wavelet has equal positive and negative areas, resulting in an average of zero. The Wavelet transform divides our track into many small segments and is therefore perfect for analyzing short and very steep peaks compared to background noise, which is the basic structure of the waves analyzed in this discussion (see Figure 4 and Figure 5). With their localized nature, wavelets can capture both frequency and time information. Three common examples of continuous wavelets are:

- mexh (Mexican Hat): Excellent for signals with discontinuities and transients such as sudden spikes or short pulses, thanks to its effective temporal localization and ability to capture rapid variations:

$$\psi(t) = \frac{2}{\sqrt{3\sqrt[4]{\pi}}} exp^{-\frac{t^2}{2}}(1 - t^2) \quad (3)$$

- cmor (Complex Morlet): Preferable for signals with well-defined frequency components, such as sinusoidal signals or those approximated by sinusoids, capable of effectively extracting dominant frequencies:

$$\psi(t) = \frac{2}{\sqrt{\pi}} exp^{-t^2} exp^{j2\pi t} \quad (4)$$

- shan (Shannon): Useful for signals with information concentrated in well-defined frequency bands, such as signals with limited band characteristics like communication signals. Suppose be B bandwidth and C frequency:

$$\psi(t) = \sqrt{B}\frac{\sin(\pi Bt)}{\pi Bt} exp^{j2\pi Ct} \quad (5)$$

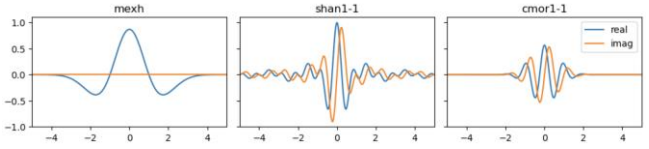

FIGURE 5. **Waves functions, showing real and image parts.**

Algorithm 3 Data augmentation of audio files

```
function AUGMENTATION(input_dir, output_dir, num_augs_per_file, noise_factor_range)
    for file_name in files in directory(input_dir) do
        if file_name ends with ".wav" then
            audio, sr ← load audio(input_dir, file_name)
            for i ← 1 to num_augmentations_per_file do
                augmented_audio ← add noise(audio, random from(noise_factor_range))
                save audio(output_dir, clean file name(file_name), augmented_audio)
            end for
        end if
    end for
end function
```

FIGURE 6. **Data augmentation algorithm pseudocode for each audio file.**

These are the main continuous wavelets and their respective mathematical wave function formulas, although many others exist, and they are the 3 wavelets that have been used to test and develop the click detection algorithm.

B. DATASET AND AUDIO SEGMENTATION

Specifically, this paragraph describes the dataset and the audio segmentation algorithm, which essentially determines a start and end point for each cut to be made for every click, calculated relative to the temporal point where the peak is found. We have collected and created four different datasets, each recorded using the microphone of a smartphone, following the methodology established by Harrison et al. in [2]. The first dataset, containing audio keystrokes from a MacBook Pro keyboard recorded via smartphone, and it has been downloaded by github [21]. The other three datasets were created on the base of first dataset but changing the speed and energy during typing on the keyboards. Each of these three datasets consists of 900 audio files. These datasets contain 75 .wav files for each of the 36 selected keys on the keyboard (letters a-z and numbers 0-9). After applying data augmentation techniques, the total number of audio files in each dataset increases to 2700. This comprehensive collection allows us to thoroughly compare the performance of various models and techniques in detecting and analyzing keystroke sounds, but using 3 different keyboards: matebook d14 keyboard, a soft keyboard and a mechanical keyboard. These datasets have been collected recording audio keystrokes with smartphones at 3 different speed type: 0.1 sec, 0.5 sec and 1 sec (see Fig. 13, 14, 15) and with different energy during the typing to create a real-world scenario. Relating to the segmentation algorithm, the start of the cut is set to the time of the peak found, while the end of the cut is set to the time of the click minus 1 second (these optimal data points were found experimentally). After each file is found, it is placed in the folder dedicated to splits, which will be our initial form of dataset.

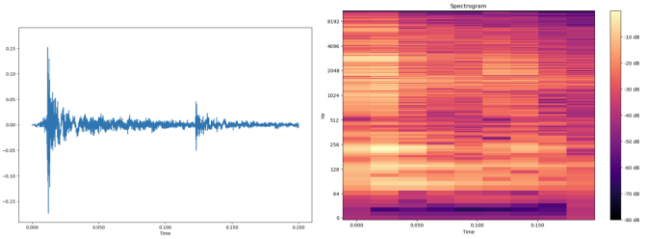

FIGURE 7. **Representation of the same audio track with intensity graph (left) and dB spectrogram (right).**

The pseudocode for the algorithm is detailed in Figure 3 as Algorithm 2: Split Audio Algorithm.

C. DATA AUGMENTATION

In this section, a description of data augmentation techniques applied to audio files. It has been applied by adding random noise and generating additional versions of the original file. The goal is diversifying and increase the size of the dataset. This process is useful for improving robustness and is commonly used in machine learning models for recognition and classification tasks, especially in contexts where the amount of available data is limited [15, 16, 17, 18, 25]. First pseudocode step is iterate through the files present in the directory of the initial dataset with segmented files, then for each audio file, a specified number of augmented versions are generated (a parameter decided beforehand and provided as an input parameter to the function) and for each augmented version, a copy of the original audio is made, and finally random noise is applied to the copied audio using the add noise function, with a low noise factor randomly extracted from a specified range. In Figure 6 there is the pseudocode.

D. TRAINING PHASE

After creating the complete dataset with data augmentation and sorting it into a CSV file, we move on to the training phase. This phase is divided into:

- feature extraction,
- construction and definition of the TCN model,
- training and evaluation,
- metrics plot for the performance.

Feature extraction step: the methods used for extracting audio features from sound signals is based on the spectrogram representation of our audio tracks [2]. These features are essential for analyzing and understanding the content of the audio signal and will be crucial for our subsequent analyses. Spectrograms are a visual representation of the content of an audio signal. They are widely used in audio analysis to detect temporal patterns. Using techniques like the Fourier transform, it is possible to transform the audio signal from the time domain to the frequency domain and then visualize it as a three-dimensional image, where the x-axis represents time, the y-axis represents frequency, and the color intensity represents the energy present at that frequency and at that

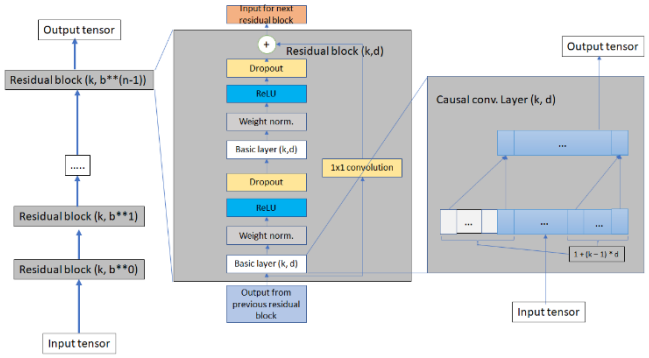

FIGURE 8. Implemented TCN model architecture.

moment. In Figure 7 an example of the conversion to a dB Spectrogram. Another method to extract audio features is using the Mel-frequency Cepstral Coefficients (MFCC) [29, 30]. MFCC are a compact representation of the spectral characteristics of an audio signal. These coefficients are calculated through a series of steps, including transforming the audio signal into the Mel scale and applying the Discrete Cosine Transform (DCT) to obtain the cepstral coefficients. MFCCs are widely used in speech analysis and recognition because they effectively capture spectral information relevant to human perception. For experiments presented in this paper it has been used Spectrograms and MFCC (see paragraph IV).

E. TEMPORAL CONVOLUTIONAL NETWORK (TCN)

This section, briefly describe the category of Temporal Convolutional Network (TCN) for the audio classification task on audio micro-details. Temporal Convolutional Networks (TCN) extend the concept of CNNs to handle temporal sequences through the use of dilated convolutional

blocks. These blocks allow the network to acquire information at multiple temporal scales, enabling greater flexibility in capturing temporal patterns at different time scales. This makes them particularly suitable for audio classification tasks, where considering the temporal relationships between signals is crucial, as in our case of signals that are difficult to characterize as simple clicks. In a TCN models, causal convolution is calculated as in (6):

$$y_i = \sum_{j=0}^{k-1} c_j x_{j-1} \qquad (6)$$

where:

- $i \in \mathbb{N}$ represents the current time index for the output signal y_i
- x_j is an input tensor at time index i,
- y_i is an output tensor at time index i,
- k is the convolution kernel,
- c_j is a convolution weight for $j \in [0, k-1]$.

The proposed TCN method, has been implemented the following causal convolution with kernel k = 3 and padding equal to k -1. To perform causal convolution, we incorporate padding on the left side of the input tensor. To execute causal

TABLE I
TABLE 1. TCN MODEL HYPERPARAMETERS

TCN Model parameters	Values
Number of layers	4
Number of classes	36
Number of filters	128
Batch size	32
Learning rate	0.001
Num channels	7
Kernel size	3
Dropout	0.2
Epochs	500
Input size	1

convolution, we employ classical 1-D convolution with padding and trim elements from the right side. Employing the dilation technique within a causal convolutional layer enhances the coverage of the input time series and substantially reduces computational costs. In the TCN architecture, it is assumed that the sequence of causal convolutional layers has a dilation factor of 2^i-1. The overall configuration of proposed TCN model architecture, is reported in Table 1. Utilizing ReLU as the activation function for TCN is recommended [1, 20]. To address potential gradient propagation issues in the hidden layers, we employ weight normalization for each convolutional layer. Additionally, dropout regularization value 0.2 is applied after every convolutional layer within the central neural network layer of TCN. In a TCN model architecture (Figure 8) everything revolves around the concept of dilation and causal convolution with input and output tensors, which ultimately are sequential data describing the information we give to the model and the conclusions we derive after processing the model. For the processing of these tensors, we must pass through various residual blocks with parameters (kernel size and dilation) formed by different convolutional layers [1]. Let's describe these concepts more precisely:

- Residual Block: The residual block is a basic unit within the TCN that helps mitigate the vanishing gradient problem during the training of deep neural networks. It consists of convolutional layers followed by a residual connection that adds the original input to the convoluted data.
- Kernel Size: The kernel size defines the size of the input window on which convolutional filters are applied. A larger kernel size captures broader temporal patterns, while a smaller size can detect finer temporal details.
- Dilation: Dilation refers to the spacing between elements of the convolution window. Dilated convolutions allow the TCN to capture temporal patterns at different scales without increasing the number of network parameters.

Algorithm 4 Prediction Algorithm

```
function PREDICT(directory_path, prefix)
    all_predictions ← empty list
    file_list ← sorted list of files in directory_path starting with prefix
    for each file_name in file_list do
        (audio, sample_rate) ← load audio file named file_name
        mfccs_features ← extract MFCC features from audio
        mfccs_scaled_features ← scale MFCC features
        predicted_label ← predict label using TCN model on mfccs_scaled_features
        prediction_class ← inverse transform predicted_label
        append prediction_class to all_predictions
    end for
    Print all_predictions as a single word
end function
```

FIGURE 9. **The prediction algorithm pseudocode.**

- Causal Convolutional Layer: The most important layer of the residual block. It is a convolutional layer that preserves the temporal order of the data during processing. It applies convolution only on past data (based on kernel size and dilation) and not on future data, which is essential for many time series tasks.

In summary, a TCN uses residual blocks, dilated convolutions, and causal convolutional layers to model temporal dependencies in sequential data, enabling better prediction or classification of future events. When implementing the TCN for audio classification on micro-details, it is necessary to define the model and training parameters adequately. In Table 1 there are the parameters used during experimental test.

F. PREDICTION ALGORITHM

The predict function takes as input the path of a directory and a prefix. This function is used to predict the word associated with the audio files present in the directory using a previously trained Temporal Convolutional Neural Network (TCN) model. The algorithm input parameters are:

- directory path: The path of the directory containing the audio files.
- prefix: The common prefix of the audio file names to be used.

Algorithm steps:

- The names of the audio files in the specified directory that start with the provided prefix are obtained and sorted based on the number extracted from the file name.
- For each audio file in the sorted list:
  (a) The audio file is loaded.
  (b) MFCC (Mel-frequency Cepstral Coefficients) features are extracted from the audio.
  (c) The MFCC features are normalized and transformed into a format compatible with the model.
  (d) Prediction is made using the TCN model.
  (e) The predicted class is decoded using the label encoder object.
  (f) The predicted word is added to the list of all predictions.
- The predicted password is printed by concatenating all the obtained predictions.

In Figure 9 the pseudocode used for the Prediction algorithm.

V. EXPERIMENTAL RESULTS

For the experiments we have been considered the same dataset splitting and all experimental settings as done by authors in the work [2] to conduct a fair comparison. Moreover, we have been executed other experiments with different keyboards, and different smartphone position in order to measure the presented model accuracy in different experimental conditions. To contrast overfitting, we have pursued an in-depth examination of phenomenon, prioritizing the model's generalization capability predicting keystroke in unseen data. Its core strength lies in the capability to capture and process sequential data, dynamically adapting to the variations in typing speed, rhythm, and inter-key intervals. This is a first step of larger research comprising benchmarking of several architectures and models. The experimentation part, has been inspired by an article dating back to August 2023 [2]. This article deals precisely with the acoustic attack capable of recognizing which keys have been clicked based on the sound emitted by the pressure of individual keys. The main limitations identified in their study are: the static splitting of audio tracks for dividing the entire file into individual click (25 clicks) files, a very high cleaning of the output audio files, and the artificial intelligence model used. CoAtNet, is a type of CNN that combines Convolution and Attention. The experimentation phase is divided into two sets of experiments with two very different tasks, generalization, and maximization. With our first phase of experiments, it has been tried to bring the system into a more ideal scenario by attempting to maximize the performance of the entire pipeline and the TCN model used, in order to achieve the state of the art regarding Keystrokes attacks. Whereas in the second phase, it has been studied the attack in more real-world contexts (wild), and therefore, we will focus on generalizing the entire process and seeing its physical limitations. In the experiment there will demonstrate that there are models that perform better under the same conditions and that slightly lower results can be achieved by significantly increasing the complexity and confusion of the data (a characteristic that distinguishes true real-world cases of potential attacks). Comparative tests have been performed with state-of-the-art models and results are shown in Figure 10, Table 2 and Table 3.

G. FIRST SERIES OF EXPERIMENTS - LAPTOP KEYBOARD IN NON-IDEAL CASES

As briefly described in the section above, this series of experiments and tests could be perfectly described in the concept of generalization. In fact, we have just described the

TABLE 2. EXPERIMENTAL RESULTS ABOUT COATNET AND THE IMPLEMENTED TCN

Model	Precision	Recall	F1-Score
CoAtNet [2]	0.960	0.950	0.950
TCN [1]	**0.980**	**0.983**	**0.983**

limitations of experiment conducted in paper [2], and in this phase, we will address all those limitations by trying to eliminate or at least improve them. This phase consists of 6 experiments. In short: 1 experiment will focus on the use of a new CNN model (experiment no. 1), another experiment will focus on generalizing the recording context (experiment no. 2), 2 experiments will study the hardware and software limitations of audio capture (experiments no. 3, 4), and finally, 2 experiments will focus on the recognition and segmentation modes of audio tracks (experiments no. 5, 6).

H. EXPERIMENT NO. 1 - TCN MODEL APPLIED TO SNOOPING KEYSTROKES ATTACK

Let's start by saying that the treatment of this experiment has been described and transposed in study [29]. Begin the experiment by analyzing the context in which experiment in [2] was conducted: an iPhone 13 placed 17 cm to the left of a MacBook Pro. Clicks are made manually at intervals of 1 second from each other. The CNN model used is CoAtNet. Initially, we only change the training pipeline by using Temporal Convolutional Networks (TCN) instead of Attentional Convolutional Networks (CoAtNet). We use the same dataset so that the results can be properly compared, and in our TCN test, we reduce the epochs to 500 for practicality. In Table 2 the results. From the observed results, it can be seen that indeed, across all fronts, the new implemented model performs better. It increases precision (correctly identified positive instances compared to all those identified as positive), sensitivity/recall (correctly identified positive instances compared to all actually positive instances in the dataset). Additionally, there are graphs showing the trend of accuracy and loss during training (see Figure 10), providing a visual demonstration of the improvement in results compared to Harrison's research paper [2] and showed in Table 3. Additionally, an analysis can be performed on the unrecognized keys, and we were able to demonstrate that keys mistakenly identified are indeed close to each other, thus making the errors more understandable. This suggests the model's effectiveness in distinguishing between different acoustic keypress classes. Such precision attests to the model's ability to capture subtle distinctions in acoustic data, highlighting its potential in enhancing security measures. It is interesting to focus on the few misclassification cases to better understand the remaining challenges. The TCN outperforms CoAtNet in keystroke recognition due to its superior ability to capture long-range dependencies and temporal patterns in sequential data. TCN's architecture, which includes causal convolutions and dilation, allows it to effectively handle the sequential nature of audio signals, making it better suited for

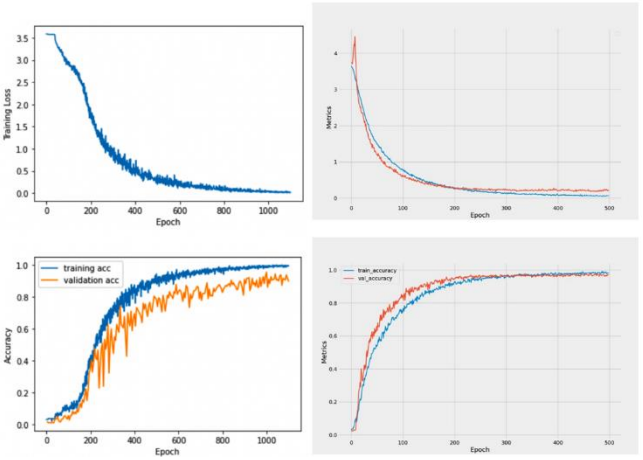

FIGURE 10. Plot on left shows experimental results applying CoAtNet (loss and accuracy), on the right the same metrics obtained with implemented TCN (train blue and validation red).

FIGURE 11. Misclassified keystroke proximity. Arrows connect the true keystroke with the corresponding misclassified one predicted by the model.

TABLE 3. OVERVIEW OF EXISTING RESEARCH PAPERS AND COMPARISON WITH OUR WORK.

Method	Accuracy validation peak
TCN (our)	**98.3%**
Harrison et al. [2]	95%
Anindya et al. [10]	93.7%
Compagno et al. [24]	91.7%
Bai et al. [6]	91.2%
Abhishek et [23]	74.3%
Zhu et al [22]	72.2%

recognizing keystrokes. TCN are superior as they inherently handle sequential data, making them adept at recognizing patterns in time-dependent signals like keystrokes. Their structure is designed for temporal tasks, ensuring efficient learning of temporal dependencies without the overhead of attention mechanisms, which may not be as beneficial for this specific application. When combined with wavelet transforms for dynamic audio segmentation, TCN can accurately isolate and classify keystrokes even in the presence of background noise and irregular typing patterns, leading to better experimental results compared to CoAtNet (a CNN with attention). The errors described in the Figure 11 are the recognition of "1" as "2," recognition of "3" as "E," recognition of "D" as "E," recognition of "M" as "4," recognition of "O" as "5," and recognition of "S" as "D".

I. EXPERIMENT NO.2 - DATASET FROM 100CM, 50CM, AND 17CM WITH LAPTOP

In this experiment, we will analyze another aspect of the limitations highlighted in work [2], which is the restricted mode of recording the audio tracks. They describe the experiment by placing the recorder at 17 cm (i.e., very close) to the keyboard, and the keys are clicked at regular intervals of 1 second (unnatural typing interval). Indeed, we wondered what would happen in a real scenario, and there would be two variables:

- Faster typing: the intervals were set at 0.1 seconds, 0.5 seconds, and 1 second between clicks
- Distance from the keyboard: three distances were chosen to provide a more general idea of the model's behavior under different and real circumstances. The distances considered are 100cm, 50cm, and 17cm.

The experiment aims to create 9 different datasets from the combinations of these 3 distances with these 3 intervals. The datasets were created using an iPhone X as the recorder with the default iOS app "Voice Memos" and a MateBook D14 (2020) for the keyboard's laptop. The laptop is positioned (see Figure 12) with the screen facing the phone, which faces the laptop's base. The measurements generated files in the m4a format, which were subsequently converted to the wav format following the following specifications:

- Audio codec: pcm_s16le
- Audio bitrate: 320kbps
- Audio channels: stereo 2.0
- Sampling frequency: 48000 Hz
- 36 audio files were generated for each data collection, each containing the pressure of a letter or number key on the keyboard 25 times.
- Data collections were performed with different distances of the smartphone from the keyboard: 17 cm, 50 cm, 100 cm.

The difference between the tracks at 100 cm and 50 cm is not very remarkable; however, graphically, we can see a thinning of the background noise level, and in general, the peaks become higher, making them more easily distinguishable in the case of a peak recognition algorithm, which we will analyze later. The most noticeable differences are observed not so much in the track with 1-second intervals but in the track with 0.5 seconds intervals, where the peculiar shape of the sought-after wave can be discerned, and especially in the track with 0.1 seconds intervals, where the audio is more recognizable as a series of clicks rather than a confused sound where it's hard to distinguish where one click ends and another begins (see Figure 13, 14, 15). Before seeing the results of the recording at 17 cm, analyzing the spectrogram and the click's patterns, and the difference between a click recognized by these recordings in wild cases compared to tracks recorded in an ideal environment as presented by Harrison et al. [2]. In Figure 13, 14, 15, it can be noted how our generalization study is proving to be correct as the two audios are significantly different in terms of readability. The last audio appears to have

FIGURE 12. Experimental desktop setup with a laptop MateBook at 17 cm

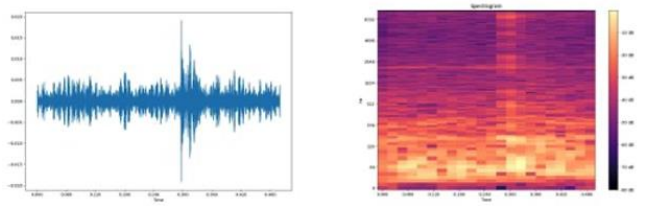

FIGURE 13. Audio track and spectrogram: Matebook D14 and iPhone X at a distance of 100 cm with a typing speed of 1 second.

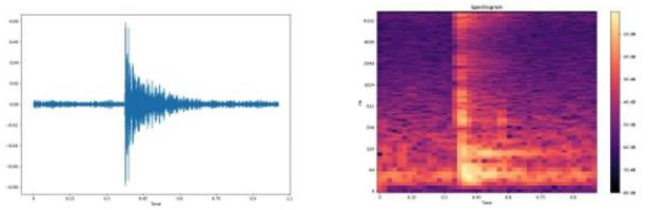

FIGURE 14. Audio track and spectrogram: Matebook D14 and iPhone X at a distance of 50 cm with a typing speed of 1 second.

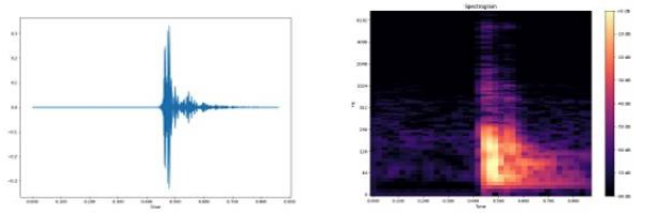

FIGURE 15. Audio track and spectrogram: Macbook Pro and iPhone 13 at a distance of 17 cm with a typing speed of 1 second as in [2].

TABLE 4. OVERVIEW OF TCN MODEL PERFORMANCE ON VARIOUS DATASETS (ALL WITH 1-SECOND TYPING INTERVALS)

Distance (cm)	Peak accuracy (%)	Peak loss
17	99.44	0.0624
50	98.88	0.0923
100	97.02	0.1244
17 (dataset used in [2])	98.30	0.1943

no background noise, and its analysis by the model will perfectly extract its features, whereas on the left, we find a more challenging but decidedly more realistic case. Finally, we have the tracks at 17 cm where we can already imagine a very low background noise, more similar to the cleaner waves analyzed in [2]. These last tracks closely resemble the result found in [2], and in fact, even in the tracks recorded at 0.1 seconds, the audio is very understandable. In the track

TABLE 5. RESULTS OF PEAKS FOUND IN THE COMPARISON BETWEEN LOSSY AND LOSSLESS ENCODING.

Coding	Key "0"	Key "9"	Key "Q"	Key "W"
Lossless (0.1 sec)	24/25	25/25	25/25	25/25
Lossy (0.1 sec)	25/25	24/25	25/25	25/25
Lossless (1 sec)	25/25	26/25	25/25	25/25
Lossy (1 sec)	24/25	25/25	26/25	25/25

TABLE 6. RESULTS OF THE INFERENCE TESTS COMPARED TO AUDIO DATASET FROM DIFFERENT DIRECTIONS.

Audio Direction	Peak Accuracy (%)
East	99.44
South	40.02
West	20.45
North	37.81

recorded at 1 second, peaks reach amplitudes 7 times higher than those at 100 cm and 50 cm, and the background noise is even lower. To conclude the experiments, demonstrate that even by generalizing the recording environment, our model performs very well, we apply the TCN model we studied to test the accuracy and loss on our 3 datasets (plus the reference of applying our model in the same dataset used in [2]). The results obtained are showed in Table 4.

J. EXPERIMENT NO. 3 - TEST OF COMPRESSED AND UNCOMPRESSED AUDIO

The objective here is to understand, in the process of generalizing the attack context, the software limitations of recording audio tracks. Briefly describe how audio is digitized and then analyze and understand the two types of audio compression: lossy and lossless. During audio recording, the analog signal from a microphone is converted into a digital signal through a process called sampling. The analog signal is measured at regular intervals in time and converted into discrete digital values. Subsequently, each sample is approximated to a digitally representable discrete value, in a process called quantization. Finally, the quantized samples are encoded using an audio encoding format that represents the digital audio data in a form that can be stored and reproduced by digital devices. Lossless compression is a type of data compression that reduces the size of an audio file without loss of quality. This type of compression is reversible, meaning that the audio data can be fully restored to its original form without loss of information. On the other hand, lossy compression reduces the size of the audio file by eliminating some information considered less relevant to human listening. This leads to a permanent loss of data and lower audio quality compared to the original. Our tests focused on understanding whether differences are also observed at the level of track analysis for feature extraction. The tests were conducted with 25 clicks performed at a distance of 17 cm for the following keys: 0, 9, Q, W. Measurements were taken both with clicks spaced 1 second apart and with clicks spaced 0.1 seconds apart. Lossy compression used was MP3, while lossless compression was WAV. From the tests showed in Table 5, it is evident that there are no significant differences between the compressed and uncompressed tracks. Some minor differences may arise due to the recordings not being identical, but the 16 tracks were recorded separately. Therefore, it appears that the compression applied does not affect the parameters we are able to analyze with our audio waveform research and study tools.

K. EXPERIMENT NO. 4: TEST FOR MULTI DIRECTIONAL AUDIO SOURCES

In this experiment, it has been conducted a trial on the physical limits of recording by changing the position of the smartphone on the desk relative to the keyboard. During presented study on generalization, it's kind of a doubt regarding the dependence of the recorded sound on the direction of the sound source, so it has been conducted some experimental tests. Physically, sound is directional, and therefore, the same sound changes its characteristics according to the listener if it originates from a different position in space relative to it. The functioning of smartphone microphones follows physical laws, and as listeners of the sound phenomenon, they are also dependent on the direction of the source. However, we still conducted tests to see if our dataset recorded in the modes described by Figure 14. In terms of waveform, there isn't any change. However, when we extract the features and compare them with our dataset calibrated for a specific direction, the accuracy results plummet drastically.

L. EXPERIMENT NO. 5 - DYNAMIC SPLIT AND WAVELET STUDY FOR IDEAL CASES

This experiment addresses the limitation of static splitting found in Harrison's treatment [2]. Static peak splitting has a major limitation because it makes dependent on typing our password to be recognized at a fixed and unnatural typing frequency. Furthermore, with static splitting, false peak problems can occur due to human imprecision. Indeed, the password is typed by a human who may not click exactly once every second, resulting in clicking a key a bit earlier and thus having the audio cut, which should represent a single click, corrupted. This phenomenon is called overlapping, and it's a serious problem because during feature extraction, the model will perceive information that doesn't belong to the click but rather as part of the pattern to learn to recognize that click. Moreover, in research [2] splitting is based on exact static number of clicks which is 25 for every key. Hence, the study of Wavelets as an analysis tool for audio tracks to identify the moment when a peak is encountered in the track and cut it from the track. Initially, the peak detection algorithm contained three variables:

- Window noise calculation: Represents the time window (in seconds) from which the background noise level is calculated. It represents the first x seconds where nothing has been clicked yet to recognize that noise as background noise.
- Peak recognition scale: Multiplicative factor to set the peak recognition threshold relative to the background noise level.

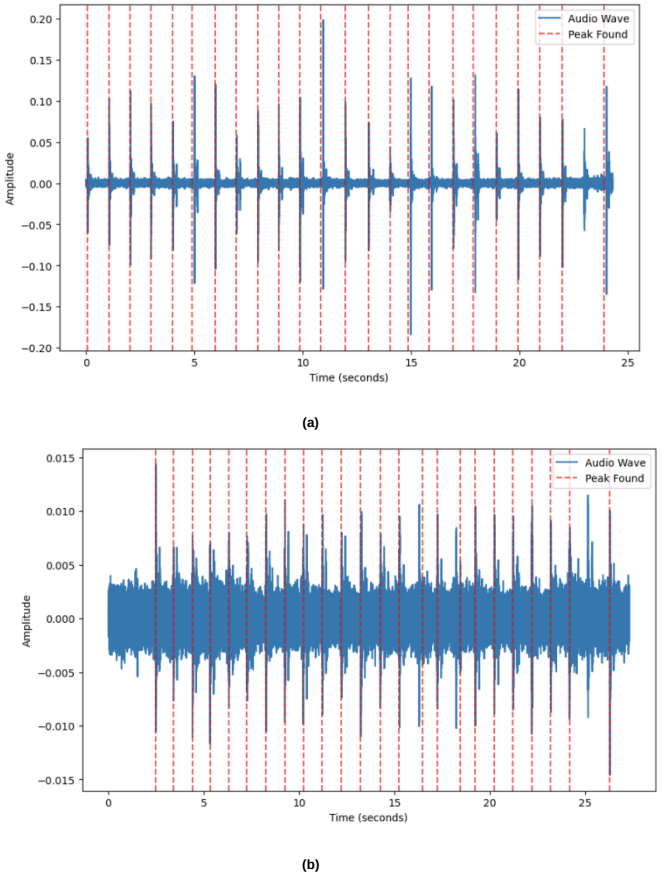

FIGURE 16. Laboratory environment audio wave with low background noise and high peaks (a) and on the wild "real world" environment audio wave with high background noise and short peaks (b).

Compared to the background noise, what can be considered a peak? A sound x time louder.

- Click time: Represents the duration of the "click" caused by pressing the key, used to skip forward in the signal after finding a peak.

Using continuous wavelet is essential to skip ahead because otherwise, it would find dozens and dozens of peaks in each click (depending on the sampling). Follow the tests carried out on different types of wavelets to see which ones perform better in our various cases. The tests are done with a distance of 17 cm and at two typing intervals, 0.1 seconds and 1 second. There are different types of wavelets that fall into two main categories: continuous and discrete. We can even disregard the discrete wavelet type because in our case, we analyze tracks to be as continuous as possible, given that we are analyzing sounds from the real world [27]. Figure 16 shows of peak detection using mexh wavelet transforms.

M. EXPERIMENT NO. 6 - PERFORMANCE TEST OF WAVELETS ON TRACKS WITH SEVERE BACKGROUND NOISE (WORST-CASE SCENARIO)

Experiment results has shown that mexh (see paragraph IV) is the most versatile, simple, and accurate wavelet in an optimal scenario, but it has limitations. Conducting tests with tracks

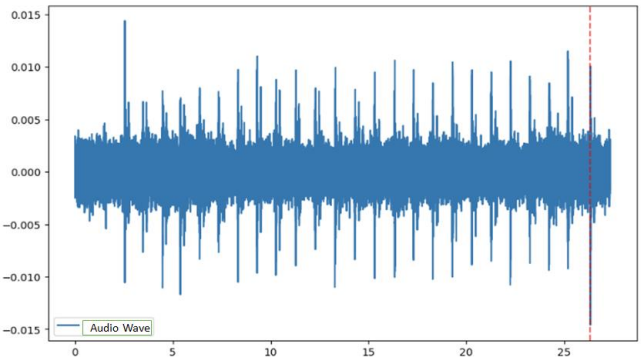

FIGURE 17. Peak recognition by the mexh Transform with a track with high background noise (peaks recognized: 1). The dashed vertical red line, identifies the unique peak detected in an audio wav with 25 clicks.

TABLE 7. RESULTS OF THE ADAPTATION TESTS OF THE SHAN B-C TRANSFORM ON A TRACK OF 25 CLICKS WITH INTERVALS OF 1 SECOND WITH HIGH BACKGROUND NOISE.

Wavelet type	Threshold	Click detection ratio
shan1.0-1.0	0.0013423	37/25
shan1.5-1.0	0.0024838	27/25
shan2.0-1.5	0.0013383	39/25
shan1.5-1.3	0.0014362	35/25
shan1.7-1.0	0.0030474	22/25
shan1.6-1.0	0.0027379	25/25

with high background noise, the effectiveness of mexh drops drastically. In that case, more complex and customizable wavelet transforms like shan (see paragraph IV) make much more sense, which, in the ideal scenario without being tuned with the right parameters, performed really poorly. This experiment, therefore, aims to compare the performance of mexh to shan in a generalized and non-ideal audio case. In Figure 17 it is showed how mexh performs in a non-ideal scenario. So now, on the same track, we need to perform tests by changing the parameters of shan B-C, where we recall B represents bandwidth and C represents frequency, in order to adapt shan to the shape of these waves (Table 7). So, by setting shan1.6-1.0, we achieve correct peak detection in a less-than-ideal scenario. Finally, to conclude this experiment, performance comparisons were calculated for the two Wavelet Transforms mexh and shan1.6-1.0 with different typing frequencies in a high background noise scenario. Moreover, experimental results, shows that application of wavelet transforms for audio segmentation mitigate the problem of different speed and energy during typing (see Figure 18).

N. EXPERIMENT NO. 7 - 17CM DATASET ON MECHANICAL KEYBOARD ON WILD CONDITIONS

This experiment has been divided into 3 parts: the description of the new data collection context, the update of the peak detection algorithm, and the results of the model applied to the new dataset. As already mentioned, the tools for this experiment are changing. Indeed, the recordings are made with an Android smartphone (Xiaomi Redmi Note 9S), and

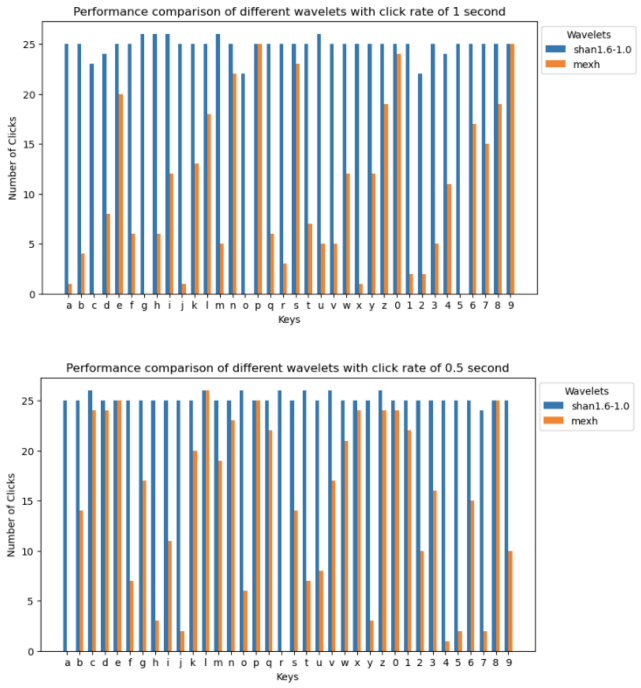

FIGURE 18. **Performance comparison of different wavelets transforms with click rate of 0.5 second and 1 second for audio wav files with 25 clicks.**

FIGURE 19. **Experiment desktop setup of the mechanical keyboard.**

the keyboard used as the source of key sound is an Akko 306B Plus ISO mechanical keyboard, with Akko CS jelly pink switches with the following construction characteristics:

- Type: Linear
- Operating Force: 45gf ± 5gf
- Total Travel: 4.0 ± 0.3mm
- Pre-Travel: 1.9 ± 0.3mm

The recorded audio files are always in WAV format with pcm s16le codec, 320kbps bitrate, stereo 2.0 audio channel, and a sampling frequency of 48kHz. The dataset to be created will consist of only 25 clicks per key with keystrokes spaced 1 second apart for the 36 alphanumeric keys on the keyboard. The keyboard and the recorder will be placed at a distance of 17 cm, and the recorder will be positioned to the left of the keyboard as in Figure 19. In Figure 20 the wave plot shows that, there is background noise. Furthermore, this detail demonstrates its natural origin, as we are certain that nothing has been intentionally or unintentionally altered by any audio cleaning software or denoiser. Before delving into the performance results of the model, examine how the detection

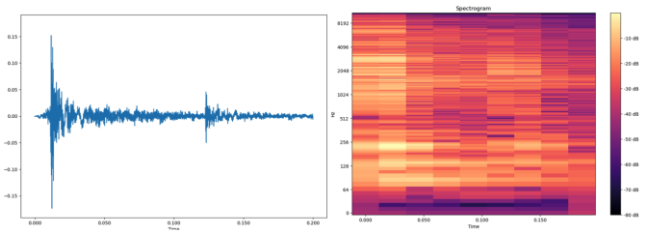

FIGURE 20. **Plot representation of the mechanical keyboard key press as a waveform (left) and as a spectrogram (right)**

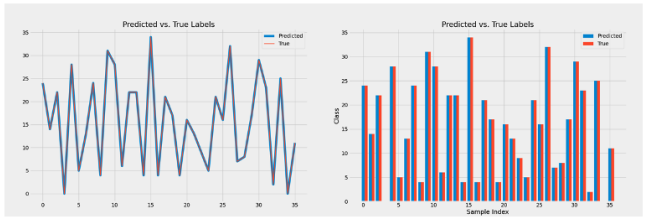

FIGURE 21. **Line (left) and bar (right) graphical representation of the comparison between predicted results and actual results. Section from 35 epochs.**

algorithm has evolved from static to semi-dynamic to dynamic. Briefly reviewing the phases of the peak detection algorithm:

- The version applied in [2, 29] wasn't exactly an algorithm but rather a static version of splitting based on the exact keystroke frequency. Issue: overlapping.
- In experiment No. 5, three parameters were introduced to make the algorithm less static (window noise calculation, peak recognition, and click time). More versatile to adapt to very different waveforms but still dependent on some recording conditions.
- In experiment No. 6, it was possible to eliminate the window noise calculation variable by calculating the background noise as half the average of the entire track, and peak recognition was set by adjusting the multiplier based on the keystroke frequency. Issue: not entirely dynamic because an approximate keystroke frequency still needs to be input. Pro: it works with any typing speed as long as you roughly know what it is.
  Now, in this latest phase of the algorithm, it's possible to be almost independent of keystroke frequency. Indeed, you can type at any speed as long as there is at least a 0.4 second gap between keystrokes or more. This result can be achieved precisely because we are in an ideal scenario.

The latest version of the algorithm is presented in the paragraph 3.A, Figure 3 under the name "Algorithm 1 Peak Detection and Split Algorithm." Now, all the experimental training data follows to evaluate the model's performance applied to this ideal scenario. The graphs are in order: line graph (predictions vs. actual values), bar graph (predictions vs. actual values) in Figure 21 and confusion matrix, accuracy trend, loss trend, classification report in Figure 22. In Figure 21 and Figure 22, experimental results show some interesting

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2025.3536877

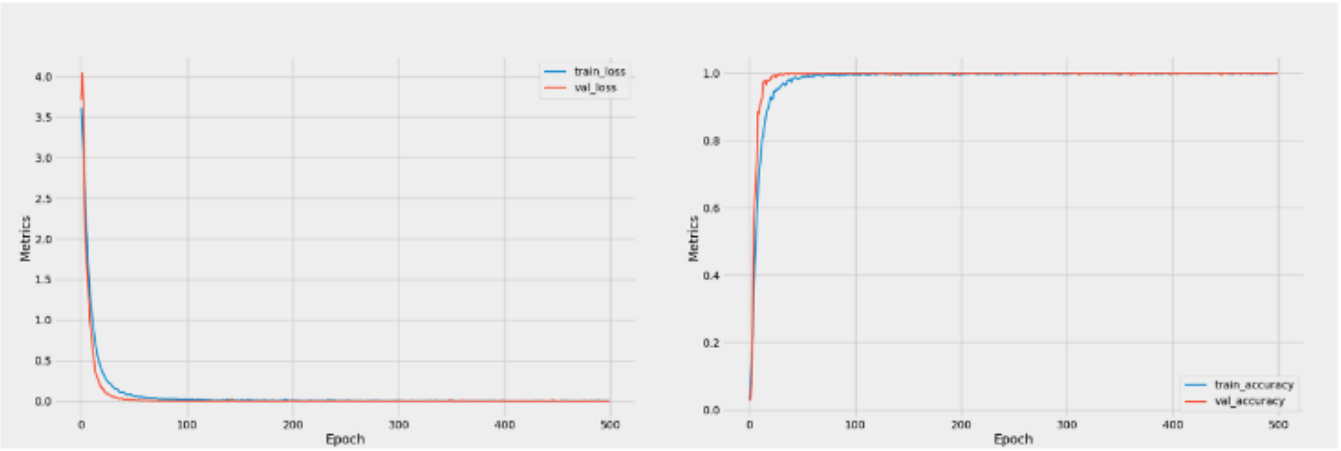

FIGURE 22. **Graphical representation of the loss trend (left) and accuracy trend (right) of the model with a mechanical keyboard.**

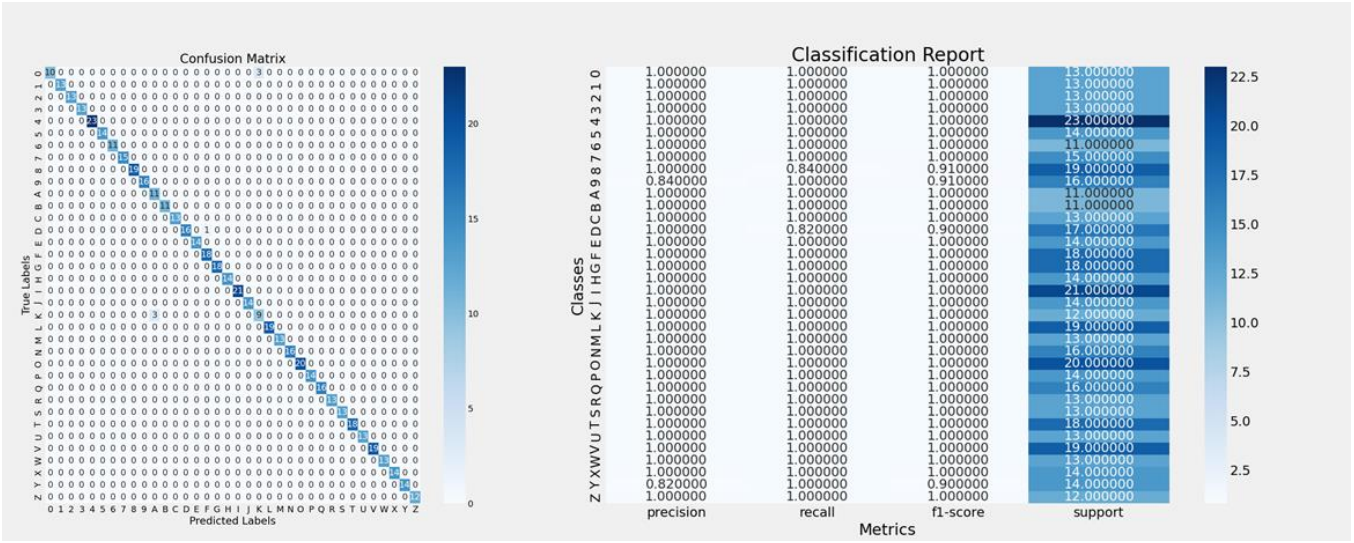

FIGURE 23. **Graphical representation of the confusion matrix related to inference tests on the trained model (left) and its summary of some metrics on various classes (right).**

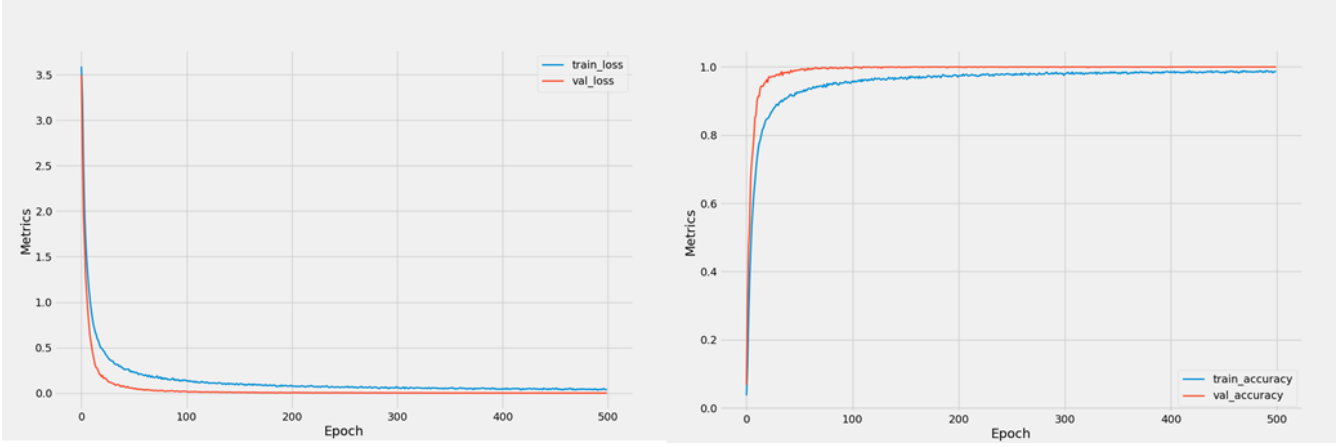

FIGURE 24. **Graphical representation of the loss trend (left) and accuracy trend (right) of the model with a soft keyboard.**

TABLE 8. RESULTS OF INFERENCES TESTS ON UNSEEN DATA FOR THE PRETRAINED TCN MODEL ON A MECHANICAL KEYBOARD.IN REAL WORLD SCENARIO

Password	Prediction	Accuracy
tesirusso2024	mesirusso2024	92.3%
tesirusso2024	tesigussp2024	84.6%
testnumero1	tesrm7mero1	72.7%
testnumero2	tesmnumero2	90.9%
15koalablu	15koalablu	100%
cupertino2090	cupermino2090	92.3%
orwell1984	prwell1994	80%
budapesthotel	budapesthotel	100%

results. In the comparison graphs between predicted labels and true labels, we observe a seemingly perfect match. Furthermore, in the training history, we can see that after approximately 100 epochs, our model already achieves very high accuracy percentages. The same applies in the opposite direction for the trend of loss, which decreases to very low values close to zero.

These data might lead us to think that our model is affected by a phenomenon called overfitting, meaning our model appears not to be learning to generalize features for prediction but rather seems to be memorizing the input data. From graphs in Figure 23, we can confidently state that our model, achieving an Accuracy of 99.00% and a Loss of just 0.0003, exhibits the phenomenon of overfitting. Performance metrics such as precision, recall, and F1 score were chosen to evaluate the model for their ability to measure precision which focuses specifically on the accuracy of positive predictions, calculating the proportion of true positive predictions out of all positive predictions (true positives and false positives), completeness (recall), and the balance between them (F1 score), ensuring comprehensive assessment of the model's effectiveness in accurately identifying and differentiating keystrokes. Figure 24 shows loss and accuracy for the same setup, showed in Figure 19 with a soft keyboard model. Overfitting occurs when the model fits too closely to the training data, memorizing not only the general patterns present in the data but also the noise and specific characteristics of the training data. Consequently, the model may lose the ability to generalize properly to new data and may exhibit lower performance when exposed to unseen data. This phenomenon occurs especially in cases where the dataset is very small, as in our case: 75 clicks for each class where only 25 are captured wild and the others are augmented. However, before raising the white flag, it is prudent to conduct inference tests on unseen data to assess how the model performs.

O. EXPERIMENT NO. 8 - INFERENCE TEST ON UNSEEN DATA

This serves as the final test to determine whether our model, despite having a clear overfitting issue, can still be valid in predicting passwords recorded at a later time compared to the dataset recording, while remaining under the same recording conditions, of course. Although overfitting may raise concerns about the model's generalization ability, it's important to consider that in certain cases, an overfitted model can still be valid if it can produce accurate results on unseen data. If the overfitted model still shows significant accuracy on unseen data, this suggests that the model is capable of generalizing correctly and making accurate predictions on new data, despite overfitting on the training data. We have 8 passwords as an example, and as showed in Table 8, the model performs quite well with an achieved mean accuracy of 89.1%, considering what one might expect from an overfitted model. Of course, there is still room for improvement, and the limitations are known and described beforehand (same smartphone recording distance and position that during acquisition of dataset). With such a small dataset, we cannot expect too much, but once again, that 89.1% of mean accuracy on inference of unseen data for an overfitted model demonstrates its quality, validity, and cutting-edge performance. In Table 3 the experimental results demonstrate a significantly contributes to the existing knowledge by advancing the field of acoustic side-channel attacks. It introduces an innovative pipeline for keystroke snooping that surpasses previous methods in accuracy and robustness. This pipeline utilizes wavelet-based dynamic audio segmentation to precisely isolate individual keystrokes, even amidst irregular typing and background noise, and employs a TCN for enhanced key classification. Rigorous testing under realistic conditions highlights the effectiveness of this approach, offering valuable insights into understanding and mitigating this evolving security threat.

P. ABLATION ON THE PROPOSED PIPELINE EXPERIMENTAL RESULTS

To isolate the contributions of the proposed individual pipeline components, it has been conducted two ablation experiments on datasets recorded in real-world scenarios: changing the energy and speed during typing and with a background noise of an empty room.

In the first ablation experiment it has been removed the smart segmentation of the recorded audio with wavelet transforms, leaving all other blocks of the pipeline unchanged. Changing the pipeline and removing the dynamic smart keystroke detection, and varying the typing speed and energy on the keyboard, returns a keystroke detection failure in the recorded audio and therefore it was impossible to carry out a classification of the audio keystrokes. The only way to apply a static keystroke detection on a recorded audio track is press every key at the same speed and energy, as done by authors in study [2, 21]. But this is not a real world-scenario.

In the second ablation experiment, it has been changed another block of the proposed pipeline, changing the features extraction method from MFCC to Mel-Spectrograms and leaving all other blocks of the pipeline unchanged. In Figure 25, 26 have been reported accuracy and loss, in Figure 27 the inference between predicted and real values in a histogram plot, and in Figure 28 the precision, recall, F1-score metrics, and the confusion matrix. These experimental results, compared to the proposed pipeline, show a lower validation

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2025.3536877

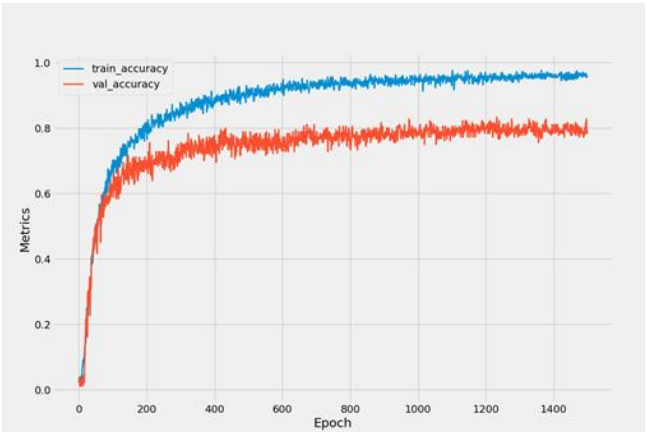

FIGURE 25. Graphical representation of the accuracy trend of the model with a Mel-Spectrograms features extraction.

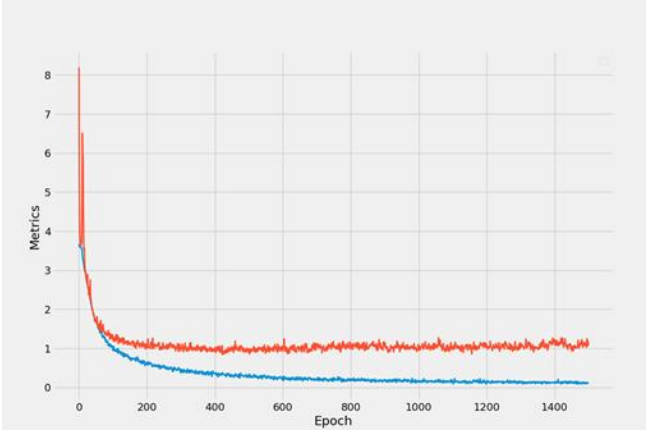

FIGURE 26. Graphical representation of the loss trend of the model with a Mel-Spectrograms features extraction

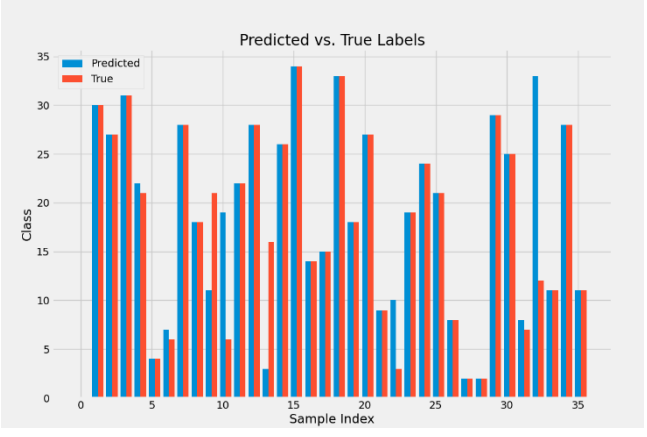

FIGURE 27. Histogram plot representation of the inference trend between predicted and true values with a Mel-Spectrograms features extraction

peak accuracy (82%) and a higher validation error loss (1.17) and these values highlight the inference errors in Figure 27 where some classes show differences between predicted and real values. Plots on Figure 28 have been demonstrated a lower precision, recall and F1-score and higher errors in each keystroke class on confusion matrix.

Q. POTENTIAL DEFENSE MECHANISMS AGAINST KEYSTROKES ATTACK

Simply typing keys in a room with a high-level ratio of background noise respect to the click audio signal, is an excellent defense strategy against Snooping Keyboard attacks. Although not explicitly suggested as a defense mechanism, imply that simply changing one's typing style could be enough to evade attacks. When touch typing was utilized, as noted in work [9] observed a reduction in keystroke recognition accuracy from 64% to 40%. While this accuracy is still notable, it may not be sufficient to accurately interpret complex inputs involving the shift key, backspace, and other non-alphanumeric keys. Moreover, altering typing style can be combined with other mitigation techniques discussed in various studies and does not require any additional software or hardware. Another straightforward defense against such attacks is using randomized passwords that include multiple cases. Given the effectiveness of language-based models as noted in study [7], passwords composed of full words may be more vulnerable to attacks. Furthermore, although several methods were able to recognize the press of the shift key, none of the surveyed papers succeeded in identifying the 'release peak' of the shift key amid the sounds of other keys. A deeper exploration of keystroke acoustics, particularly the 'release peak' (the sound made when a key is released) can significantly impact model performance. Understanding these subtle acoustic features helps improve attack accuracy. While pressing a key produces distinct sound patterns, the 'release peak' is less pronounced and often masked by other keystrokes, complicating recognition. Analyzing these release peaks alongside press sounds can refine models to better capture and differentiate these acoustic signatures, ultimately enhancing both attack precision and the development of robust countermeasures. This focus could bridge gaps in existing studies and lead to more effective keystroke detection and defense mechanisms. In order to solve this issue in the proposed pipeline it has been applied the wavelet transforms to smart detect signals of key press and key release (see Section IV). This doubles the potential search space for characters following a shift key press. The authors in research [37] tried to interfere with the acoustic characteristics of keystrokes by randomly altering the sound slightly whenever keystrokes were detected. This approach reduced the accuracy of FFT features to the level of random guessing, but it only slightly affected the MFCC features.

Keystroke attacks pose significant security risks as they exploit the unique acoustic or behavioral patterns of typing to infer sensitive information. The threat extends beyond individual privacy breaches to organizational vulnerabilities, especially in environments where high-value data is entered manually. Attackers can deploy these methods remotely using smartphone’s microphones, making detection and prevention challenging. However, the generalizability of such attacks is limited by several factors. Variations in keyboard models, typing styles, ambient noise, and recording conditions can

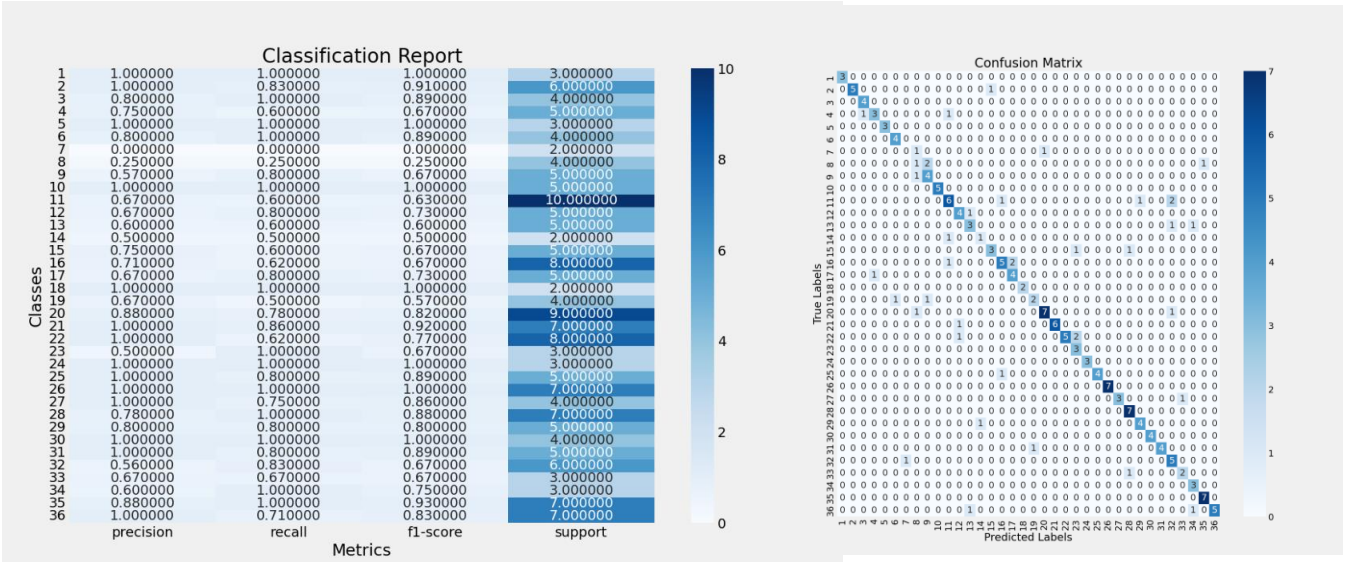

FIGURE 28. Precision, recall and F1-score for all key classes with a Mel-Spectrograms features extraction (left) and confusion matrix of the model with a Mel-Spectrograms features extraction (right)

reduce the accuracy of these methods. Additionally, attackers must overcome encryption methods and other security measures implemented in modern devices. While advancements in machine learning, such as TCN and wavelet-based audio preprocessing, have improved recognition rates, these technologies also highlight a potential gap between controlled experimental conditions and real-world application scenarios. Real environments are far noisier and more variable, potentially limiting the effectiveness of trained models. Addressing these challenges requires robust adversarial training and contextual awareness models to minimize false positives and negatives. Despite these limitations, the increasing sophistication of keystroke attacks underscores the need for stronger countermeasures, such as typing pattern obfuscation or acoustic shielding, to mitigate the risks posed by these evolving threats.

Future research could focus on automatically suppressing or removing keystroke sounds from VoIP applications, enhancing both security and user experience. In work [14], two-factor authentication (2FA) is recommended, utilizing secondary devices or biometric checks to access data. As more laptops incorporate biometric scanners, the need for keyboard input diminishes, reducing the risk of Acoustic Side Channel Attacks (ASCAs). Nonetheless, as [14] points out, other data besides passwords remain at risk. Interestingly, some previously effective countermeasures have become less viable over time. For instance, paper [4] suggested that touchscreen keyboards, being silent, could negate ASCAs. However, compromised smartphone microphones have since shown a concerning ability to infer text typed on touchscreens with high accuracy. Similarly, the recommendation to check for microphones in a room before typing sensitive information is now impractical due to the ubiquity of microphones in modern devices like smartphones, smartwatches, laptops, and smart speakers. The advice from study [24] to mute microphones or avoid typing during Skype calls has also become less feasible, especially with the increase in remote work and video conferencing during the COVID-19 pandemic. The growing prevalence of technology that facilitates these attacks raises concerns about the sufficiency of current countermeasures.

VI. CONCLUSIONS AND FUTURE WORKS

Throughout this research, we have succeeded in developing a keystroke prediction model with unprecedented accuracy compared to the methods previously explored by the scientific community. Our innovative approach is based on the use of Temporal Convolutional Networks, integrated with a dynamic keyboard click recognition algorithm. This combination has proven to be extremely effective in creating an advanced system to support an individual's privacy data attack. Generalizing the study of the problem, we have also maximized its potential by bringing our high-performing model even to ideal cases. Furthermore, it is a significant achievement to have brought such a high-performing and complex model to smartphones, demonstrating that espionage can also be carried out simply by pressing a button in an application created for this purpose. This raises concern as such attacks becomes potentially easier to execute and more accessible. The accuracy of the predictions made by the model discussed in this paper, especially in non-ideal scenarios and real-world contexts, raises awareness of the danger of the attack and draws attention to a real vulnerability regarding the category of acoustic password espionage attacks. Research in this field is still in development and offers significant potential for the future. With the increasingly widespread use of

artificial intelligence, it is likely that attack systems will become more sophisticated. our dataset to address the issues mentioned and provide more comprehensive results. In our future studies, we plan to expand our dataset with adversarial testing to address the issues mentioned and provide more comprehensive results. Finally, it is important to emphasize that this discussion is by no means intended to promote or encourage attacks of this kind, but rather to highlight a vulnerability often overlooked in the cybersecurity landscape.

REFERENCES

1. Hatice Vildan Dudukcu, Murat Taskiran, Zehra Gülru Çam Taskiran, Tulay Yildirim: Temporal Convolutional Networks with RNN approach for chaotic time series prediction. Appl. Soft Comput. 133: 109945 (2023).
2. J. Harrison, E. Toreini and M. Mehrnezhad, "A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards," in 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Delft, Netherlands, (2023) pp. 270-280.
3. ZihangDai, HanxiaoLiu, QuocV.Le, MingxingTan "Coatnet: Marrying convolution and attention for all data sizes". In: Advances in Neural Information Processing Systems 34 (2021).
4. Dmitri Asonov and Rakesh Agrawal. "Keyboard acoustic emanations". In: IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004. IEEE. 2004, pp. 3–11.
5. Backes, M., Dürmuth, M., Gerling, S., Pinkal, M., & Sporleder, C. 'Acoustic Side-Channel Attacks on Printers'. USENIX Security Symposium. (2010).
6. Jia-Xuan Bai, Bin Liu, and Luchuan Song. "I Know Your Keyboard Input: A Robust Key-stroke Eavesdropper Based-on Acoustic Signals". In: Proceedings of the 29th ACM Inter-national Conference on Multimedia. 2021, pp. 1239–1247.
7. Yigael Berger, Avishai Wool, and Arie Yeredor. "Dictionary attacks using keyboard acoustic emanations". In: Proceedings of the 13th ACM conference on Computer and communications security. 2006, pp. 245–254.
8. Jeffrey Friedman. "Tempest: A signal problem". In: NSA Cryptologic Spectrum 35 (1972), p. 76.
9. Tzipora Halevi and Nitesh Saxena. "Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios". In: International Journal of Information Security 14.5 (2015), pp. 443–456.
10. Maiti, A., Armbruster, O., Jadliwala, M., & He, J. Smartwatch-Based Keystroke Inference Attacks and Context-Aware Protection Mechanisms. Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. (2016)
11. NSA NACSIM. "5000: Tempest Fundamentals". In: National Security Agency (1982).
12. Ehsan Toreini, Brian Randell, and Feng Hao. "An acoustic side channel attack on enigma". In: School of Computing Science Technical Report Series (2015).
13. Peter Wright. "Spycatcher: The Candid Autobiography of a Senior Intelligence Officer". In: New York: Viking (1987).
14. Li Zhuang, Feng Zhou, and J Doug Tygar. "Keyboard acoustic emanations revisited". In: ACM Transactions on Information and System Security (TISSEC) 13.1 (2009), pp. 1–26.
15. Park, D.S., Chan, W., Zhang, Y., Chiu, C., Zoph, B., Cubuk, E.D., & Le, Q.V. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Inter-speech. (2019).
16. Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: Journal of big data 6.1 (2019), pp. 1–48
17. Huq, S., Xi, P., Goubran, R., Valdés, J.J., Knoefel, F., & Green, J. (2023). 'Data Augmentation using Reverb and Noise in Deep Learning Implementation of Cough Classification'. IEEE International Symposium on Medical Measurements and Applications (MeMeA), 1-6. (2023).
18. Yashish M. Siriwardena, Ahmed Adel Attia, Ganesh Sivaraman, Carol Espy-Wilson "Audio Data Augmentation for Acoustic-to-articulatory Speech Inversion using Bidirectional Gated RNNs." arXiv:2205.13086 (2022).
19. Gridin I., "Time Series Forecasting using Deep Learning: Combining PyTorch, RNN, TCN, and Deep Neural Network Models to Provide Production-Ready Prediction Solutions" (2021).
20. Wang, H., & Zhang, Z. (2022, May). TATCN: time series prediction model based on time attention mechanism and TCN. In 2022 IEEE 2nd international conference on computer communication and artificial intelligence (CCAI) (pp. 26-31). IEEE.
21. J. Harrison, Keystroke-Datasets https://github.com/JBFH-Dev/Keystroke-Datasets (2023)
22. Zhu, T., Ma, Q., Zhang, S., & Liu, Y. 'Context-free Attacks Using Keyboard Acoustic Emanations'. Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. (2014).
23. S Abhishek Anand and Nitesh Saxena. "Keyboard emanations in remote voice calls: Password leakage and noise (less) masking defenses". In: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy. 2018, pp. 103–110.
24. Compagno, A., Conti, M., Lain, D., & Tsudik, G. 'Don't Skype & Type!: Acoustic Eaves-dropping in Voice-Over-IP'. Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. (2016).
25. Kim, G., Han, D.K., & Ko, H. "SpecMix: A Mixed Sample Data Augmentation method for Training with Time-Frequency Domain Features." Interspeech (2021).
26. Taheritajar, A., Harris, Z.M., & Rahaeimehr, R.: A Survey on Acoustic Side Channel At-tacks on Keyboards. ArXiv, abs/2309.11012 (2023).
27. Thomas, D.R., Drishya, V., Rajeshwari, G.R., Sundar, L., & Prabhu, V. (2022). A discrete wavelet transform-based audio watermarking technique for digital security. International Journal of Nonlinear Dynamics and Control.
28. Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. IEEE Trans. Inf. Theory, 36, 961-1005.
29. Spata, M.O., Ortis, A., Battiato, S., Russo, V.M. (2024). A New Deep Learning Pipeline for Acoustic Attack on Keyboards. In: Arai, K. (eds) Intelligent Systems and Applications. IntelliSys 2024. Lecture Notes in Networks and Systems, vol 1065. Springer, Cham. https://doi.org/10.1007/978-3-031-66329-1_26
30. M.O. Spata, V.M. Russo, A. Ortis, S. Battiato, Acoustic Side Channel Attack for Keystroke Splitting in the Wild. To be published in IEEE MetroXrain 2024 conference proceedings.
31. Fan, J., Zhang, K., Huang, Y., Zhu, Y., & Chen, B. (2023). Parallel spatio-temporal attention-based TCN for multivariate time series prediction. Neural Computing and Applications, 35(18), 13109-13118.

MASSIMO ORAZIO SPATA graduated in 1998 in Computer Science, received PhD degree in Computer Science from Dipartimento di Matematica ed Informatica in 2008 discussing the thesis entitled "Tecniche basate sull'Equilibrio di Nash per lo scheduling in un Cluster Grid". He joined STMicroelectronics Catania in 1999 as Distributed System Engineer. In November 2009 he joined STMicroelectronics Technical Staff Engineer Group, developing know how and skills in System Integration and Image Processing. Since January 2013 he has joined the STMicroelectronics R&D group where developing know how and skills in biomedical portable devices for DNA analysis. Since 2017 he has been teaching Computer Science at secondary schools where developing know how and skills in robotics systems. Since January 2022 he is a research fellow in Computer Science at the University of Catania, Dipartimento di Matematica e Informatica where developing know how and skills in deep learning algorithms for biomedical, audio and biometric application. He collaborates with Universities and Research Institutions involved in different research projects. Author of different patents and publications for journals and international conferences concerning the above fields.

VALERIO MARIA RUSSO holds a Bachelor's degree in Computer Science from the University of Catania, where he completed a thesis focused on the application of artificial intelligence in audio recognition. During his studies, he developed skills in Android app development, integrating deep learning technologies to enhance user interaction with the word around. Through his thesis work, he had the opportunity to engage in scientific research, co-authoring articles with his professors. He is currently aiming to continue his studies with a Master's degree in Artificial Intelligence.

ALESSANDRO ORTIS received the master's degree (summa cum laude) in computer science from the Universitá degli Studi di Catania, in 2015, and the Ph.D. degree in mathematics and computer science from TIM, in 2019. His Ph.D. thesis investigates several aspects related to visual sentiment analysis applied on crowd sourced images/videos. He has been involving in the field of computer vision research, since 2012, when he joined the Image Processing Laboratory (IPLab). He did two research internships with STMicroelectronics, in 2011/2012 and with TIM, in 2015. He is currently a Postdoctoral Researcher with the Universitá degli Studi di Catania. He is the coauthor of 15 articles published in international conferences and four journals and co-inventor of one international patent. His current research interests include computer vision, machine learning, and multimedia. He received the Archimede Prize for the excellence of academic career conferred by the University of Catania, in 2015. He is a reviewer of several international conferences and journals..

SEBASTIANO BATTIATO received the degree (summa cum laude) in computer science from the University of Catania, in 1995, and the Ph.D. degree in computer science and applied mathematics from the University of Naples, in 1999. From 1999 to 2003, he was the Leader of the "Imaging" Team, STMicroelectronics, Catania. He joined the Department of Mathematics and Computer Science, University of Catania, as an Assistant Professor, an Associate Professor, and a Full Professor, in 2004, 2011, and 2016, respectively. He is currently a Full Professor of computer science with the University of Catania, where he is also the Scientic Coordinator of the Ph.D. Program in computer science. He is involved in the research and directorship with the Image Processing Laboratory (IPLab). He coordinates IPLab's participates on large scale projects funded by national and international funding bodies and private companies. He is also the Director (and the Co-Founder) of the International Computer Vision Summer School (ICVSS). He has edited six books and coauthored about 250 articles in international journals, conference proceedings, and book chapters. He is a co-inventor of 22 international patents. His current research interests include computer vision, imaging technology, and multimedia forensics. He has been a Regular Member of numerous international conference committees. He was a recipient of the 2017 PAMI Mark Everingham Prize for the series of annual ICVSS schools and the 2011 Best Associate Editor Award of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He has been the Chair of several international events, including ICIAP 2017, VINEPA 2016, ACIVS 2015, VAAM 20142016, VISAPP 20122015, IWCV 2012, ECCV 2012, ICIAP 2011, ACM MiFor 20102011, and SPIE EI Digital Photography 2011-2013. He has been a Guest Editor of several special issues published in international journals. He is an Associate Editor of the Journal of Electronic Imaging (SPIE) and the IET Image Processing Journal.