

HW-Spy: Handwriting Inference by Tracing Pen-Tail Movements

Long Huang
University of Michigan
Ann Arbor, USA
huanlong@umich.edu

Kang G. Shin
University of Michigan
Ann Arbor, USA
kgshin@umich.edu

Abstract

While keyboard typing has been the most common way of inputting texts, handwriting still plays an important role in generating, inputting, or recording information like filling out essential/private forms. Considerable research has been done to identify and demonstrate the risk of keystroke-inference attacks. However, little has been done on handwriting inference despite its high risk of leaking sensitive information. To assess this under-explored risk of information leakage, we present a novel handwriting-inference attack, called HW-Spy, by tracing the victim's pen-tail movements when both the pen tip and the writing surface are outside the view of the attacker's camera, which usually happens when the victim is multitasking during an online meeting, when the victim's writing scene (in a public space) is recorded by a remote camera, or when the victim's writing behaviors are captured by the surveillance camera in a bank/dealership/realty office.

In particular, we apply image segmentation to the recorded video frames of the victim's writing activities and extract the victim's pen-tail movements as a 2D coordinate sequence. We then identify the stroke-associated movements from the recorded pen's in-air video frames using a 1D U-Net model trained for stroke mask prediction and segment the characters based on the thus-derived motion features. The pen-tail's coordinate segments are then fed into a Long Short-Term Memory (LSTM) network to reconstruct the actual handwriting, which is processed further by a transformer-based model to infer the hand-written content. Our extensive experimentation shows HW-Spy to achieve an accuracy, up to 84.2%, of personalized handwriting inference and a comparable accuracy, up to 79.5%, of non-personalized handwriting inference.

CCS Concepts

• Security and privacy;

Keywords

Handwriting; Pen Tail; Image Segmentation; Transformer

ACM Reference Format:

Long Huang and Kang G. Shin. 2025. HW-Spy: Handwriting Inference by Tracing Pen-Tail Movements. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25)*, October 13–17, 2025, Taipei, Taiwan. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3719027.3765109>



This work is licensed under a Creative Commons Attribution 4.0 International License. *CCS '25, Taipei, Taiwan*

© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1525-9/2025/10
<https://doi.org/10.1145/3719027.3765109>

1 Introduction

As keyboard typing has become the most common way of inputting texts, there have been numerous studies of keystroke inference using various sensors to capture keystroke dynamics. However, very few of them have investigated handwriting inference despite its still important role in generating and editing sensitive documents, such as filling out tax/employment/application forms. Such handwriting activities could be spied on by the adversary to infer the victim's private information, including but not limited to, name, address, phone number, Social Security Number (SSN), annual income, and bank account numbers, which poses a serious threat to personal information security and privacy. It is, therefore, important to thoroughly investigate the risks of handwriting inference.

State-of-the-art (SOTA) handwriting-inference attackers try to infer the handwritten content using various side channels. For example, the motion sensors embedded in a smartwatch are used to capture the victim's hand movements during writing [45], or a microphone was placed next to the writing surface to record the writing sound [12, 51, 52]. Similarly, a magnetic sensor was placed next to the victim to measure the magnetic signal changes caused by a stylus pen during writing [26]. The network traffic was also monitored to estimate/infer the handwriting on a touch-screen [20]. Recently, radio frequency (RF) signals were utilized to detect handwriting through a wall [56]. However, all these attacks require dedicated hardware or sensors to be placed very close to the victim or even attached to the victim's body parts, thus making them infeasible in practice. In sharp contrast, HW-Spy aims to infer the handwritten content from the video of the victim's pen-tail movements recorded by a commonly available webcam during online meetings, by a smartphone/tablet camera placed remotely from the victim, or by a surveillance camera hacked/installed by the adversary in a bank/dealership/realty office, requiring neither dedicated hardware nor close proximity to the victim and the writing surface. A similar idea was proposed by Patil *et al.* [32],¹ and a pilot study was conducted. However, unlike HW-Spy, it relies on a limited dataset and uses simple motion tracking and classification algorithms, thus achieving much lower performance. Note that although there are acoustic channels in the online meetings, the aforementioned handwriting inference via audio side-channels would not work due to the online meeting platforms' embedded noise suppression/cancellation, which automatically removes the echo and the sound other than the human voice in the audio.

When people are writing sensitive information on paper or device screen (e.g., filling out bank account/credit card information or social security number) in a public space, they may always check if there is anyone nearby or if there is a camera(s) aiming at the writing surface. They may also use the non-writing hand or other

¹This unpublished work was brought to our attention by a reviewer.

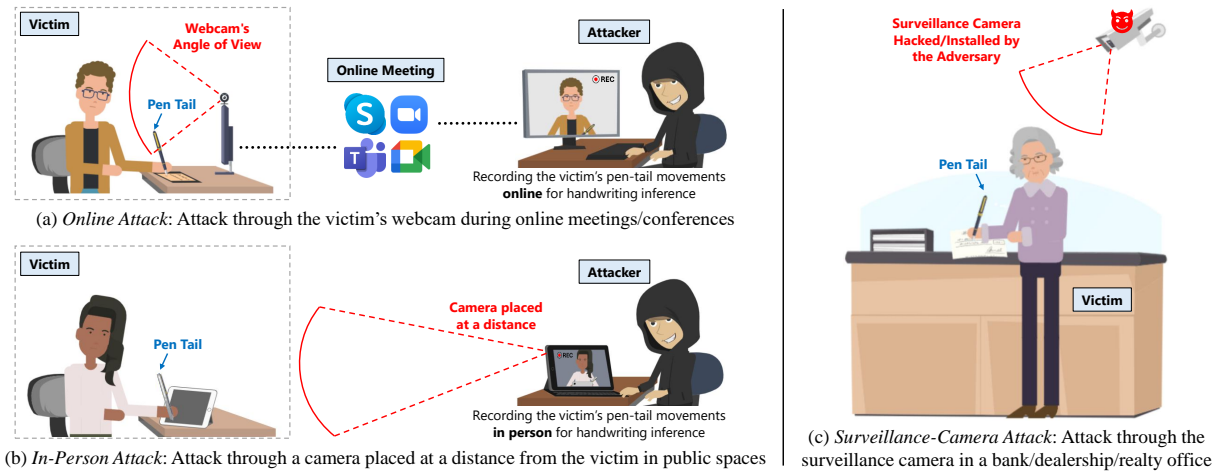


Figure 1: Attack scenarios of HW-Spy.

parts of their body to cover the writing surface in order to prevent shoulder surfing. However, the movements of the victim's pen-tail – that may still carry sensitive information – can be recorded with a distant camera or a surveillance camera and exploited by the attacker to infer the content written by the target/victim. Similarly, multitasking has proved to be a common behavior during online meetings [8]. People tend to perform other tasks when they are not speaking, such as filling out forms on paper or touchscreen. Since the writing surface (e.g., a paper or tablet screen) is always outside the webcam's angle of view, people tend to believe there would not be any eavesdropper of information they write/input, even with their webcams turned on during the meeting. However, the movements of the tail of their pen, pencil, or stylus pen can still be captured and recorded through the victim's webcam by the attacker present in the same meeting.

The attacker's goal is to infer the handwritten content from the victim's pen-tail movements captured by a distant camera in a public space, by the surveillance camera in a bank/dealership/realty office, or by a webcam turned on during an online meeting. The attacker first extracts the victim's pen-tail movements as a 2D coordinate sequence by applying image segmentation to the recorded video frames of the victim's writing activities. The attacker then identifies the stroke-associated movements from the pen's in-air movements using a 1D U-Net model trained for stroke mask prediction and segments the characters based on the derived motion features. The attacker next develops an Long Short-Term Memory (LSTM) network to reconstruct the victim's actual handwriting from the segmented pen-tail coordinates. The attacker further develops a transformer-based model to learn the characteristics of different written characters and predict their labels. Specifically, this paper makes the following contributions:

- Proposal of a novel handwriting inference attack by tracing the victim's pen-tail movements in the video recorded with a commonly available webcam during online meetings, with a smartphone/tablet camera remotely from the victim in a public space, or with a surveillance camera in a bank/dealership/realty office. The proposed attack requires neither dedicated hardware

nor close proximity to the victim and the writing surface, thus facilitating its feasibility and deployment;

- Development of novel algorithms to
 - (1) Track the victim's pen-tail movements: computer vision methods are explored and used to trace the pen-tail movements in the recorded video frames and extract the pen-tail coordinates;
 - (2) Perform stroke mask prediction and character segmentation: we develop a 1D U-Net model to identify the stroke-associated movements from the pen's in-air movements and derive motion features from pen-tail movements to segment characters;
 - (3) Reconstruct the actual handwriting and infer the handwritten content: we develop an LSTM network to reconstruct the actual handwriting from a sequence of pen-tail coordinates. We also develop a transformer-based model and leverage its attention mechanism to learn the characteristics of different letters & symbols, and predict their labels;
- Building expanded datasets: we extend the data collection to cover not only the upper- and lower-case English alphabets but also Arabic numerals, and special characters, which are usually used in account IDs and passwords;
- Improvement of inferring handwritten content: Our results show that HW-Spy not only outperforms the SOTA by achieving a higher inference accuracy but also closes the gap between personalized and non-personalized handwriting inferences.

2 Related Work

2.1 Keystroke Inference Attacks

Keystroke-inference attacks have been extensively studied in the past where the attacker eavesdrops on the victim's keyboard-striking activities using different side channels. Existing keystroke inference attacks can be classified based on the (vision, sound, motion, wireless, and electromagnetic) sensors they use.

Vision-based attacks often require to track the victim's hand movements or touching fingertip by placing a camera above or in front of the victim [5, 24, 38, 50, 53, 54]. This can also be achieved by capturing the reflection on the victim's eyeballs or glasses [34, 48]. Recent studies show that keystroke information can be leaked

through the victim’s eye movements [9, 44] or upper body movements during a video call [37]. However, these vision-based attacks are limited by the lighting condition and the camera’s field of view. Sound-based attacks record the typing sound using a microphone placed next to the keyboard [4, 59, 60] and extract the acoustic features that are key-specific for inferring keystrokes. However, these attacks rely largely on the quality of key-striking sound, and are not always feasible in the noisy real-life environments.

Motion-sensor-based attacks leverage the accelerometer and gyroscope embedded in a smartwatch worn by the victim to track its hand movements for inferring keystrokes [25, 27, 42, 43]. Moreover, the typing-induced vibration can be captured by an accelerometer placed close to the keyboard [28]. Wireless-sensor-based attacks leverage the WiFi devices placed next to the victim and analyze the channel state information (CSI) of the WiFi signals interfered with by the typing behavior for inferring the victim’s keystrokes [2, 13, 18, 23, 49, 55]. Electromagnetic-sensor-based attacks attach the sensor underneath a table to measure the human-coupled electromagnetic emanations from the touchscreens to infer the victim’s input on a mobile device [19]. However, these methods require either malware installation or physical proximity to the victim/device.

2.2 Handwriting Inference Attacks

Unlike the extensively studied keystroke inference attacks, there have been only a few studies exploiting handwriting inference. The hand movements captured by the motion sensors embedded in the smartwatch can be used to infer the handwritten content on paper, whiteboard, or in the air [3, 45–47]. Acoustic signals can be actively sent and reflected to track the hand movements for handwriting inference [31, 57]. Acoustic sensors placed next to the writing surface can passively record the writing sound [12, 51, 52], which is then used for stroke detection and written letter recognition. The handwriting with the stylus pen on a touchscreen can induce specific magnetic patterns [26], which are captured by a magnetometer for handwriting inference. The network traffic generated during the handwriting on the touchscreen can be used to derive the timing and length of the strokes, which are then used to recover the written characters [20]. A 6 GHz frequency modulated continuous wave (FMCW) radar system has recently been developed to detect handwritten content behind a wall [56]. However, these approaches require dedicated hardware or eavesdropping devices to be either placed close to the victim or even directly attached to the victim’s hand/wrist, making them easily located and removed. In contrast, HW-Spy analyzes the victim’s pen-tail movements captured by a webcam during online meetings, by a camera placed remotely from the victims in a public space, or by a hacked/fake surveillance camera in a bank/dealership/realty office, to extract the handwriting coordinates and infer the written content. A similar idea was proposed by Patil *et al.* [32], whose threat model likewise assumes that the attacker’s camera has a view of only the back end of the victim’s pen, without direct visibility of the paper or pen tip. A pilot study involving a single test subject writing digits 0–9 was conducted. The authors employed an optical flow-based method to track the pen’s motion, whose paths were compared with the transformed ground truth paths using Time Warp Edit Distance (TWED) [29]. The digit whose path yields the lowest TWED score is selected as

the predicted class for the test sample. In contrast, HW-Spy adopts image segmentation to track the pen-tail movements, which proves to be more accurate than their optical flow-based approach. Moreover, rather than simply transforming the ground truth path to compare with the pen’s path—which heavily depends on precise camera pose estimation—HW-Spy trains an LSTM network for handwriting reconstruction, providing greater robustness. Additionally, HW-Spy utilizes a transformer model for handwriting inference, which significantly outperforms their distance-based classification. Although there are also acoustic channels in online meetings, none of the above-mentioned sound-based attacks works due to the online meeting platforms’ built-in noise cancellation, which automatically removes the echo and the sound other than the human voice from the audio.

3 System Overview

3.1 Threat Model

We consider the attack scenarios where an adversary (i) spies on the writing activities of online meeting/conference participants with their (web)cameras turned on, (ii) shoulder-surfs people’s writing activities in public spaces using remote camera(s), or (iii) monitors the customers’ writing activities through the surveillance cameras in a bank/dealership/realty office. In particular, we consider the following (realistic) attack scenarios.

- **Online Attack:** As illustrated in Fig. 1(a), the target/victim is assumed to be multitasking during an online meeting/conference, writing some sensitive information with a pen/pencil/stylus (such as filling out a tax or employment form). Since the writing surface (e.g., a paper/tablet on the table) is always outside the (web)camera’s angle of view, the victim tends to believe that there would be no spying on his/her hand writing. However, the (web)camera could capture the victim’s pen-tail movements. By recording and then analyzing the victim’s pen-tail movements, the adversary could infer the handwritten content.
- **In-Person Attack:** In the scenario illustrated in Fig. 1(b), the victim is assumed to be writing on a paper/touchscreen in a public space like a library, cafe, airport, etc. To prevent potential eavesdropping, the victim ensures no shoulder surfer or hidden camera nearby and also covers the writing surface with his/her own body or other items. The victim then tends to believe there will be no spying on his/her hand writing. However, the victim’s pen-tail movements could still be captured with a remote camera and analyzed by the adversary to infer the handwritten content.
- **Surveillance-Camera Attack:** Fig. 1(c) illustrates the scenario in which the victim is assumed to be filling out an application form/contract/check in a bank/dealership/realty office. To prevent potential privacy leakages, the victim turns his/her back to the surveillance camera and tries to hide the writing surface. The victim then tends to believe that the written content cannot be seen by the camera. However, the surveillance camera might still be able to capture the victim’s pen-tail movements, which can be acquired by the adversary (either via hacking into the surveillance system or placing a fake surveillance camera) and analyzed to infer the handwritten content.

Depending on the type of writing media used by the victim, we further consider the following writing scenarios:

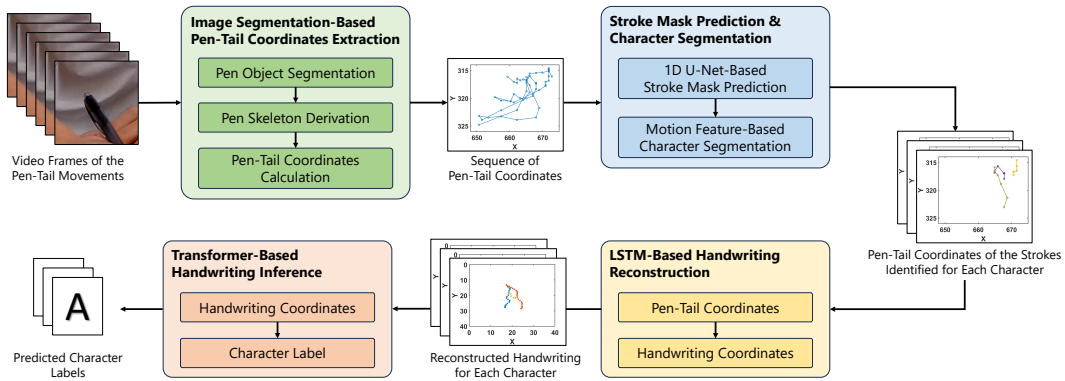


Figure 2: The architecture of HW-Spy.

- **Writing on Paper:** The victim is assumed to be writing on paper with a pen/pencil, such as filling out a tax/job-application form, a contract, or a check. The paper and the pen tip are both outside the (web)camera’s angle of view during an *Online Attack* or are both blocked during an *In-Person Attack* or during a *Surveillance-Camera Attack*, while the pen-tail movements can be captured by the (web)camera, the attacker’s camera placed remotely, or the surveillance camera in a bank/dealership/realty office.
- **Writing on Touchscreen:** The victim may also write on the screen of a tablet or a smartphone with a stylus pen when filling out some digital forms. The touch screen and the tip of the stylus are both outside the (web)camera’s angle of view during an *Online Attack* or are both blocked during an *In-Person Attack* or during a *Surveillance-Camera Attack*, while the movements of the stylus pen-tail can still be captured by the (web)camera, the attacker’s camera placed remotely, or a surveillance camera.

3.2 System Flow

Fig. 2 depicts the architecture of HW-Spy. From the pen-tail movements captured in the video of the victim’s writing activity, *Image Segmentation-Based Pen-Tail Coordinates Extraction* derives the pen-tail coordinates by applying the image segmentation to each of the video frames (which are essentially RGB images). In particular, the pen object is first segmented from the image background, resulting in a pixel area of interest. Then, the skeleton of the pen is derived from the segmented pixels using linear regression, after which the pen-tail coordinates of the current video frame are calculated by taking the top point of the derived pen skeleton. This process is repeated for all incoming video frames, yielding a sequence of pen-tail coordinates. Next, *Stroke Mask Prediction & Character Segmentation* identifies the exact strokes from the in-air movements between strokes using a 1D U-Net model trained for stroke mask prediction and derives motion features from the sequence of pen-tail coordinates for character segmentation. The segmented pen-tail coordinates are then fed into *LSTM-Based Handwriting Reconstruction* and transformed into handwriting coordinates by an LSTM network. To adapt to the input and output sizes of the LSTM network, the handwriting coordinates are down-sampled to have the same frame rate as the pen-tail coordinates. In the last step, *Transformer-Based Handwriting Inference* predicts the label for each written character

by taking the reconstructed handwriting coordinates for that character as the input. In particular, the transformer-based model first converts the reconstructed coordinates of a written character into a feature vector that is compatible with the model dimension using a linear transformation layer. Then, positional information is added to the feature vector, which helps preserve the order of the reconstructed coordinates. The resulting vector is then processed by a transformer encoder, which has N identical layers and leverages the attention mechanism to learn the characteristics of different written characters. The learned high-level representations of the character are then processed by a multi-scale Convolutional Neural Network (CNN) and further passed through a linear transformation layer and a softmax layer to predict the character label.

4 Design of HW-Spy

4.1 Extraction of Pen-Tail Coordinates

Given the recorded video of the user’s writing activities, we need to first extract the pen-tail coordinates. We tried two different extraction methods. The first is based on the optical flow estimation that has been widely used for tracking moving objects in videos [6]. The second leverages the image segmentation [30], which is capable of partitioning a digital image into multiple meaningful segments, such as different objects. These two methods are evaluated and compared in Sec. 6.1.

4.1.1 Optical Flow-Based Method. We first tried an optical flow-based method to capture the pen-tail movements, which follows the Horn-Schunck method [17] by assuming the optical flow is smooth across the entire image. Fig. 3(a) shows the estimated optical flow from two consecutive frames in the recorded video of the user’s writing activity. The optical flow estimated at the edge of the pen is more obvious than the other parts, because the intersection part of the pen and the background has larger pixel changes when the pen is moving. We thus apply thresholding to the optical flow estimated at each pixel position and obtain the contour of the pen’s upper part (as shown in Fig. 3(b)), which is then used to calculate the pen-tail coordinates.

4.1.2 Image Segmentation-Based Method. We also tried to segment/separate the pen from the background in the video frame and then calculate the pen-tail coordinates based on the segmented pen

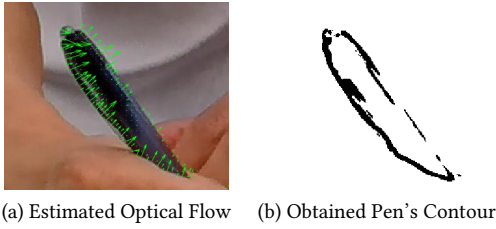


Figure 3: Extracting pen-tail movements using optical flow.

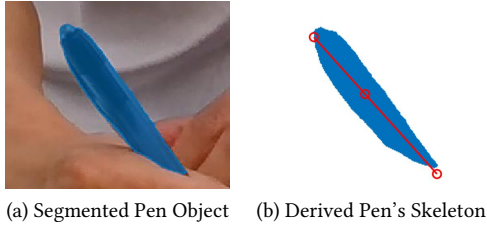


Figure 4: Extracting pen-tail movements using image segmentation.

object. Image segmentation partitions a digital image into multiple parts or regions based on the characteristics of the pixels in the image. Traditional image segmentation methods extract features from the pixels' color values and apply simple learning-based algorithms for the segmentation. Such methods include thresholding, histograms, edge detection, cluster-based segmentation, etc. [7]. Although the traditional image segmentation is cost-efficient, it cannot analyze more sophisticated images. Therefore, deep learning-based segmentation methods have been proposed [15], which are trained with annotated image datasets and can thus associate every pixel of an image with a class label.

We adopt the most recent Segment Anything Model (SAM) [21] developed by Meta to segment the pen from each video frame. SAM is composed of an image encoder built upon a pre-trained Vision Transformer [1], a prompt encoder built upon CLIP [33], and a mask decoder built upon a Transformer decoder block [41]. The model is trained with 1 billion masks on 11 million images, which can transfer zero-shot to new images. Fig. 4(a) shows the pen segmented from the video frame by SAM, which is marked with a blue mask. We can observe that the contour of the pen is captured well and the pen is clearly separated from other objects such as the user's hand and cloth. Next, we collect the coordinates of the pixels within the segmented pen and apply a linear regression to derive the pen's skeleton, which is illustrated in Fig. 4(b). The top point of the skeleton is recorded as the pen-tail coordinates for the current video frame. By processing the video frames of the user's writing activity, we can obtain a sequence of pen-tail coordinates.

4.2 Stroke Mask Prediction and Character Segmentation

After obtaining the sequence of pen-tail coordinates, we first determine which parts of the coordinates correspond to the written strokes rather than the pen's in-air movements between strokes using a 1D U-Net model trained for stroke mask prediction. Then,

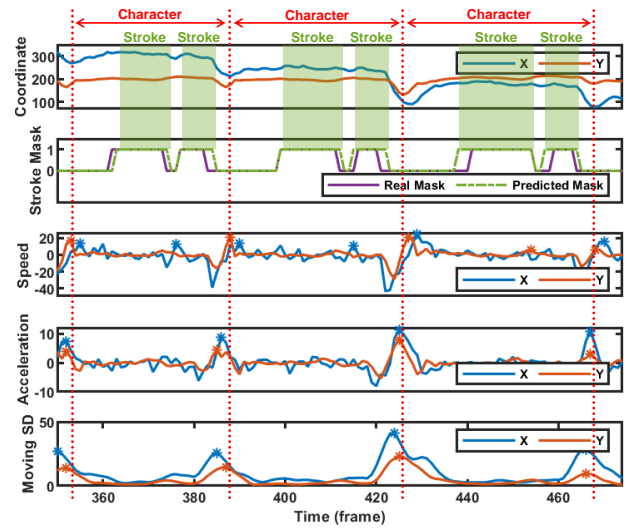


Figure 5: An example of stroke mask prediction and character segmentation.

we segment the characters based on the motion features derived from the coordinate sequence.

4.2.1 1D U-Net-Based Stroke Mask Prediction. The pen-tail coordinates captured during a handwriting activity not only correspond to the exact strokes written but also contain the in-air movements between the strokes and characters. To identify the stroke-associated movements from the pen's in-air movements, we train a 1D U-Net model to predict the mask of the strokes. In particular, we modify the original U-Net model [36] by replacing the 2D convolutional layers, transposed convolutional layers, max-pooling layers, and crop layers with their 1D versions, which yields a 1D U-Net model capable of processing sequence input. The 1D U-Net model takes the sequence of pen-tail coordinates as the input and outputs a stroke mask of the same length as the sequence, where each digit of the mask is either 1 or 0, indicating whether the coordinate sample corresponds to a stroke or not. The 1D U-Net model is trained with human-annotated stroke masks and later used for stroke mask prediction during attacks.

4.2.2 Motion Feature-Based Character Segmentation. We observe that the pen tail tends to move differently between characters than during the writing of a character (e.g., the pen tail generally moves more rapidly between characters than during the formation of individual characters. This is intuitive, as the writer need not carefully trace the strokes between characters and can instead move swiftly to the starting point of the next character). Therefore, we derive motion features from the sequence of pen-tail coordinates to segment the characters. In particular, we derive the speed, acceleration, and moving standard deviation from the sequence of pen-tail coordinates and find their common peaks. The strokes between adjacent peaks are then segmented to be one character. Fig. 5 shows an example of the predicted stroke mask and segmented characters for a sequence of pen-tail coordinates. Specifically, the strokes are first obtained using the mask predicted by the 1D U-Net model

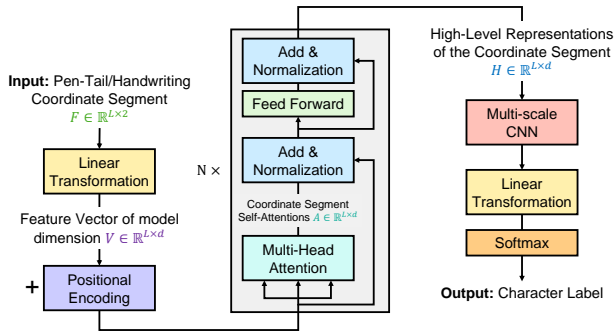


Figure 6: Transformer-based handwriting inference model.

and further split into characters based on the common peaks found between the derived motion features.

4.3 LSTM-Based Handwriting Reconstruction

The pen-tail trajectories extracted from the recorded videos of the user’s writing activities are sequences of 2D coordinates and essentially a type of time-series data. The user’s handwriting can also be viewed as a type of time-series data, where 2D coordinate sequences can be extracted from the handwriting trajectories. The process of acquiring handwriting trajectories will be described in Sec. 5.2 since they are only used as the ground truth for reconstructing the handwriting, not part of the attack process.

The handwriting reconstruction now requires the transformation of the pen-tail coordinates into the handwriting coordinates, which is a sequence-to-sequence translation. We thus leverage the LSTM network for handwriting reconstruction, which has been widely deployed for processing time-series data. Specifically, we create an LSTM network with an sequence input layer that has a feature dimension of 2, corresponding to the x and y components of the pen-tail coordinates. Followed by the input layer are three stacked bidirectional LSTM layers with 200, 400, and 800 hidden units, respectively. Then, a dropout layer with the dropout probability 0.2 is added after each bidirectional LSTM layer to prevent the network from memorizing specific training data. At the end of the network, a fully-connected layer of size 2 is used as the output layer, which corresponds to the x and y components of the handwriting coordinates. The LSTM network is then trained with the pen-tail coordinates as the input and the handwriting coordinates as the output for handwriting reconstruction.

4.4 Transformer-Based Handwriting Inference

We have developed a transformer-based model to learn the characteristics of different written characters from either the pen-tail movements or the reconstructed handwriting for inferring the written content. Transformers were initially introduced for sequence transduction [41] and have been explored extensively for processing time-series data [58] and images [11]. Recent studies find transformers outperforming CNN and RNN by processing all data samples in parallel and enabling each data sample to attend to all other samples [10, 39]. We find that transformer encoders can learn from small data segments even at a millisecond level with both spatial and temporal characteristics. Specifically, transformers process the

entire segment at once, and positional encodings are added to make the model aware of the sample index in the segment.

We developed a transformer encoder-based model to classify the labels of the written characters. The model checks the pen-tail coordinate segment or the reconstructed handwriting coordinate segment to predict the associated character label. Fig. 6 shows the structure of our transformer model. The pen-tail/handwriting coordinate segment $F \in \mathbb{R}^{L \times 2}$ of length L is taken as input, which is first applied with a linear transformation to be turned into a feature vector $V \in \mathbb{R}^{L \times d}$ compatible with the model dimension d . Next, positional encoding adds the coordinate segment’s data samples index information to the feature vector. The resulting feature vector is then passed through the transformer encoder, which is composed of N identical structures. Within each structure, a multi-head attention mechanism is applied to derive the coordinate segment’s self-attentions at each data sample. Accordingly, the transformer encoder outputs the high-level representation of the coordinate segment $H \in \mathbb{R}^{L \times d}$, describing the spatial and temporal character information carried in each segment. The high-level representation is finally passed through a multi-scale CNN followed by a linear transformation and a softmax function to estimate the character label. The trained model has 170,306 parameters and a size of less than 0.7MB. It requires 16.8M FLOPs for a single forward pass.

4.4.1 Linear Transformation & Positional Encoding. All sub-layers in the transformer model produce the output of the model dimension d to facilitate the add & norm computations. With an input coordinate segment $F \in \mathbb{R}^{L \times 2}$, we first apply a linear transformation to turn it into a feature vector $V \in \mathbb{R}^{L \times d}$ that is compatible with the model dimension. The parameters used in the linear transformation are learnable. As there is no convolutional layer or recurrent layer in the transformer model, in order to make it aware of the index/order of a coordinate segment’s data samples, we add positional encodings to the feature vector. Specifically, sinusoidal encodings are used, which are basically sine and cosine functions of different frequencies [41]. Alternatively, learnable positional encodings can also be used.

4.4.2 Transformer Encoder. The transformer encoder is formed with N identical layers. Within each layer, there is a multi-head attention sub-layer followed by a position-wise feed-forward sub-layer. A residual connection and a layer normalization are applied to each of the two sub-layers. The output of each sub-layer has the same dimension d , facilitating add and norm computations.

Self-Attention. When the model is processing one pen-tail/handwriting coordinate data sample, self-attention allows it to attend to all other data samples in the segment enabling it to learn the relationships between the coordinates at different data samples. To calculate self-attentions of the segment, we first create three vectors — the Query vector, the Key vector, and the Value vector — from the transformer encoder’s input feature. These three vectors are created by multiplying the input feature vector by three matrices that are learnable during training. Then, a score is calculated by performing the dot product of the Query vector and the Key vector. For the features at data sample t , its attention scores against the features at all data samples in the input feature vector are calculated by the dot product of the Query vector for data sample t (q_t)

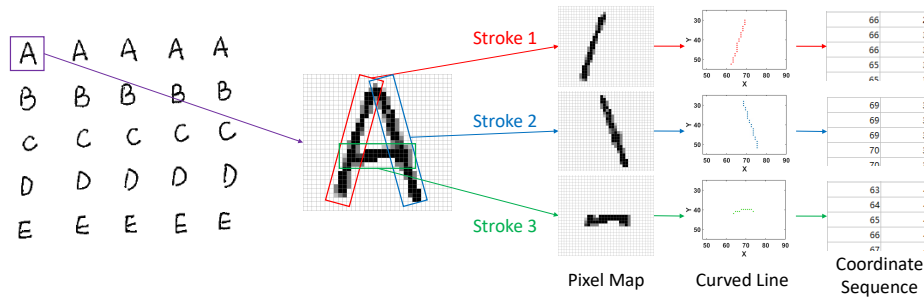


Figure 7: Ground truth acquisition of the handwriting trajectories.

and the Key vectors for all data samples (k_1, k_2, \dots, k_L). The scores then go through a softmax function to make itself add up to 1. The softmax score determines how much of each data sample’s feature information could be expressed by that of the currently examined data sample. The value vector at each data sample is then multiplied by the softmax score, and summed up to generate the self-attention for the input coordinate segment at the current data sample.

Multi-Head Attention. Rather than a single attention function, we use multi-head attention to allow the model to focus on different data samples of the input coordinate segment and give the attention layer multiple representation sub-spaces, which helps capture character-specific features. In particular, the Query vector, the Key vector, and the Value vector are linearly projected multiple times by different learnable projection matrices. The projected versions of the three vectors are used to perform the attention function in parallel, the results of which are finally concatenated and projected to generate the final attention results of the coordinate segment.

5 Experimental Setup

We evaluate HW-Spy mainly for the *Writing on Paper* scenario during an *Online Attack* whose experimental setup is described in Sec. 5 and evaluation results in Sec. 6.1, 6.2, 6.3, 7, and 9. For *In-Person Attack*, *Surveillance-Camera Attack*, and *Writing on Touchscreen* scenarios (flat or tilted), the experimental setup and evaluation results are presented in Sec. 6.4, 6.5, and 6.6 for comparison.

For the *Writing on Paper* scenario during an *Online Attack*, we conducted experiments on Zoom. The victim is assumed to be writing something with a pen during an online meeting with the (web)camera turned on. The camera’s angle of view allows only the upper part of the pen to be captured in the video. The attacker thus records the video to infer the victim’s handwriting. The proposed attack does not require the attacker or the victim to use any particular device, as long as the victim joins the meeting with a camera (either a smartphone/laptop camera or a webcam) turned on and facing towards him/herself. The resolution and frame rate of the recorded video is fixed at 1280×720 p and 30 FPS, respectively, which are supported by most commodity phone cameras or webcams and the Zoom platform. The proposed attack can be easily extended to operate on other online meeting platforms such as Google Meet, Skype, Teams, etc.

5.1 Data Collection

For the *Writing on Paper* scenario during an *Online Attack*, we recruited 20 participants (8 females and 12 males) aged from 21 to

38 for our experimental evaluation. The participants are formed by undergraduate/graduate students and company employees.² The participants sit at a regular distance (50 – 100cm) from the screen/camera and join a zoom session with the camera turned on and facing towards themselves. During data collection, the participants are asked to use their own gel pens to write characters in regular size (2 – 4mm for lowercase letters and 4 – 6mm for uppercase letters) both *separately* and *consecutively* on 8.5" × 11" letter-size paper. The videos capturing the pen-tail movements are recorded and later analyzed for handwriting inference. For each participant, we collect the data in two sessions separated by somewhere between 2 and 4 weeks. The first-session data is used for training and validation leveraging k -fold cross-validation ($k = 5$), while the second-session data is used only for testing/inference.

5.1.1 Writing Characters Separately. We first asked the participants to write characters *separately*, including uppercase and lowercase English alphabet letters, Arabic numerals, and special symbols. They were asked to repeat writing every character 20 times during each of the two sessions.

Uppercase and Lowercase English Alphabet letters are part of private/personal information such as name, address, and password. So, we ask the participants to write the 26 uppercase English alphabet letters (i.e., A-Z) as well as the 26 lowercase English alphabet letters (i.e., a-z).

Arabic Numerals are usually used in driver license numbers, phone numbers, home/office addresses, dates of birth, passwords, etc. So, the participants are asked to write the 10 Arabic numerals.

Special Symbols are often used in passwords to prevent the adversary from hacking. Some special symbols are also used frequently in our daily lives, such as using “\$” for money, “#” for numbers, and “%” for percentages. We thus identify the most frequently used 10 special symbols and ask the participants to write them, including “-”, “.”, “?”, “!”, “*”, “@”, “#”, “\$”, “%”, and “&”.

5.1.2 Writing Characters Consecutively. The participants are also asked to write characters *consecutively*, which includes writing pangrams that cover all English alphabet letters and writing sensitive information such as address, password, phone number, SSN, and Date of Birth (DoB). We also consider both *disconnected* and *connected* characters to cover different writing styles. For each scenario, the participants are asked to write the pre-selected pangrams and sensitive contents 10 times, which are only used for testing.

²IRB exemption has been obtained and the data are anonymized.

Pangrams. We search for 5 pangrams that cover all the English alphabets and ask each participant to write them. Our goal is to evaluate the handwriting inference in a more general setting that is not biased by any specific letters.

Address. The addresses are randomly generated by an online tool [14], which follows the format of US addresses.

Password. As recommended by the National Institute of Standards and Technology, passwords should be at least 12 characters long, and 14 characters or more are better [40]. We thus use an online password generator [35] to randomly generate a 14-character password for each participant.

Smartphone Number/SSN/DoB. For each participant, we also randomly generate a 10-digit phone #, a 9-digit SSN, and a DoB following the US format of writing dates.

5.2 Ground-Truth Acquisition

Although the ground truth of the written content is known, contains the labels for the written characters and can be used to evaluate the handwriting inference, we still need to acquire the ground truth of the actual handwriting (i.e., the trajectories of the written characters) on letter-size paper and use it for the evaluation of handwriting reconstruction.

In particular, we ask the participants to scan the letter-size papers they have written on using either a copier or the smartphone’s scanning function (depending on the resources available to them) and send us the scanned PDF files. We then convert the PDF files into PNG format and process them in Matlab, which are loaded as 2D pixel maps. For each character written by the participants, we first segment the strokes based on the corresponding video. Next, we apply the medial axis transform [22] to the 2D pixel maps of the segmented strokes to reduce the 2D strokes to 1-pixel wide curved lines. We then obtain the coordinates of the points in the curved lines and use them as the trajectories of the written characters.

Fig. 7 shows an example of acquiring handwriting trajectories. For the character “A” in the scanned image, three strokes are segmented, whose 2D-pixel maps are then reduced to 1-pixel wide curved lines. The coordinates of points in the curved lines are then obtained and used as the trajectories for the character’s strokes. The trajectories obtained for each written character are finally rescaled to fit into a 20×20 bounding box centered at point (20, 20), before being used as the ground truth for handwriting reconstruction (as shown in Fig. 8).

5.3 Evaluation Metrics

The attacker aims to reconstruct the victim’s handwriting from his/her pen-tail movements and infer the handwritten content, where handwriting reconstruction is a *regression* task and handwriting inference is a *classification* task. Below we introduce the metrics used for evaluating these two tasks.

5.3.1 Handwriting Reconstruction – Regression. As handwriting reconstruction is a regression, we evaluate its goodness by measuring the Mean Squared Error (MSE) between the character trajectories in the reconstructed handwriting and the original handwriting.

5.3.2 Handwriting Inference – Classification. We evaluate the accuracy of handwriting inference, representing the percentage of

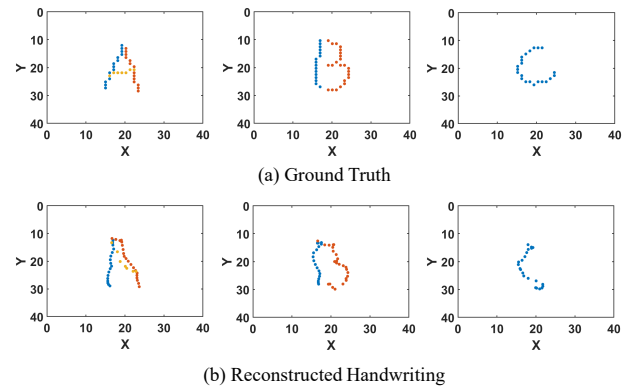


Figure 8: Illustration of the handwriting reconstruction.

the handwriting that has been correctly inferred by the adversary. Specifically, we evaluate three – character, word, and sentence – levels of accuracy.

Character-level accuracy is evaluated by measuring the percentage of the written characters whose labels have been correctly inferred by the attacker.

Word-level accuracy is evaluated by applying a spell-checker [16] to each inferred word and measuring the percentage of correctly inferred characters after spell-checking.

Sentence-level accuracy is assessed by applying a grammar-checker [16] to each inferred sentence and measuring the percentage of correctly inferred characters after spell-checking and grammar-checking.

5.4 Personalized vs. Non-Personalized

We evaluate both personalized and non-personalized performance for the proposed handwriting reconstruction and inference. In particular, depending on whether or not the adversary has access to the victim’s data during the training, s/he can train a personalized or non-personalized model for the handwriting reconstruction and inference. If the victim’s data is available during the training process, the adversary can use such data to train a personalized model tailored to that victim and perform handwriting reconstruction and inference based on the personalized model. However, in practice, the victim’s data may not be available during the training process. For example, the victim is a new person whose data has never been seen before by the adversary. In such a case, the adversary can only use the data collected from other victims to train a “non-personalized” model and then use it for reconstructing and inferring the (previously-unseen) victim’s handwriting. We present the personalized performance of handwriting reconstruction and inference in Sec. 6 and the non-personalized performance in Sec. 7.

Table 1: Performance of personalized handwriting reconstruction for separately written characters.

| Pen-Tail Coordinates Extraction Method | Avg. MSE (pixels) | | | |
|--|-------------------|-------------------|-----------------|------------------|
| | Uppercase Letters | Lowercase Letters | Arabic Numerals | Specific Symbols |
| Optical Flow | 9.1 | 14.7 | 6.9 | 8.3 |
| Image Segmentation | 6.3 | 10.2 | 3.8 | 4.9 |

6 Personalized Inference

We first evaluate the personalized performance of HW-Spy, where a separate model is trained and tested for the victim using its own data. In particular, for each of the participants selected as the victim, we use its first-session data to train and validate the 1D U-Net model, the LSTM network, and the transformer model. Training a personalized model for 100 epochs with a batch size of 32 and a learning rate of $1e^{-3}$ takes less than 20 minutes on four RTX-6000-Ada-48GB GPUs. The performances of personalized handwriting reconstruction and inference are then evaluated/tested using the victim’s second-session data. The results presented below are averaged among all the participants.

6.1 Personalized Handwriting Reconstruction

We first present the performance of the personalized handwriting reconstruction. Fig. 8 shows some examples of the reconstructed handwriting and their corresponding ground truths. One can observe that the proposed LSTM network successfully reconstructs the strokes of each character. After combining the reconstructed strokes, we can obtain a character with shape and trajectory similar to the ground truth, which can be identified by humans. The reconstructed handwriting can then be used as the input of the proposed transformer model for inferring the handwritten content.

Besides the qualitative analysis of the above handwriting reconstruction, we also compare the quantitative reconstruction performance when the pen-tail coordinates extracted by the optical flow-based method or the image segmentation-based method are used as the input. As shown in Table 1, for separately written characters, the proposed LSTM network reconstructs the upper- & lower-case letters, Arabic numerals, and special symbols with an average MSE of 6.3, 10.2, 3.8, and 4.9 pixels, when the pen-tail coordinates extracted by the image segmentation-based method are used, outperforming the MSE achieved by using the pen-tail coordinates extracted by the optical flow-based method. Similarly for consecutively written characters, the LSTM network achieves a better reconstruction performance when the pen-tail coordinates extracted by the image segmentation-based method are used, reconstructing the disconnected characters with an average MSE of 4.6 pixels and the connected characters with an average MSE of 13.8 pixels, as shown in Table 2. So, we choose image segmentation as the pen-tail coordinates extraction method for HW-Spy.

6.2 Inferring Personalized Handwriting

We next evaluate the performance of the personalized handwriting inference for both separately and consecutively written characters.

6.2.1 Separately Written Characters. We train our transformer-based handwriting inference model with both the derived pen-tail

Table 2: Personalized handwriting reconstruction performance for consecutively written characters.

| Pen-Tail Coordinates Extraction Method | Avg. MSE (pixels) | |
|--|-------------------------|----------------------|
| | Disconnected Characters | Connected Characters |
| Optical Flow | 8.0 | 17.9 |
| Image Segmentation | 4.6 | 13.8 |

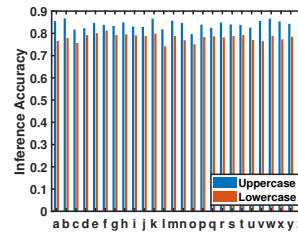


Figure 9: Performance of inferring English alphabets.

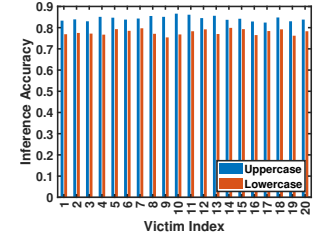


Figure 10: Inference performance for each target/victim.

coordinates and the reconstructed handwriting. We also include two conventional ML algorithms for comparison.

Using Pen-Tail Coordinates as Input. Table 3 shows the performance of inferring separately written characters using the model trained with the pen-tail coordinates. Our transformer-based model is shown to outperform SVM and KNN in inferring different types of characters. In particular, the transformer-based handwriting inference achieves accuracies of 80.3%, 75.4%, 89.5%, and 85.2% in inferring the uppercase letters, the lowercase letters, the Arabic numerals, and the special symbols, respectively.

Using Reconstructed Handwriting as Input. The inference performance for separately written characters using the model trained with the reconstructed handwriting is also provided in Table 3. The transformer-based model is shown to outperform SVM and KNN in inferring different types of characters. Moreover, using the reconstructed handwriting as the input to the model further improves the inference performance to 84.2%, 78.1%, 91.3%, and 87.5% for uppercase letters, lowercase letters, Arabic numerals, and specific symbols, respectively. These results corroborate the efficiency of our handwriting reconstruction, and hence we use the model trained with it for the rest of our evaluation.

Fig. 9 details the performance of HW-Spy in inferring the English alphabet letters. One can observe that HW-Spy always performs better in inferring the uppercase letters than the lowercase letters, because the uppercase letters are usually written with more strokes than the corresponding lowercase letters, thus revealing more user-specific writing patterns that the adversary can infer. In particular, more than a half of the uppercase letters are inferred with an accuracy above 84%, where inferring letters “B”, “K”, and “W” achieves the best performance with accuracies of 86.7%, 86.6%, and 86.6%, respectively, while the inference accuracy for “C”, “L”, and “O” degrades to 81.7%, 81.8%, and 79.7%, respectively. These results reflect the fact that “C”, “L”, and “O” only have one stroke and are similar to each other. On the other hand, most of the lowercase letters are inferred with an accuracy above 77%, where inferring letters “e”, “f”, and “k” achieve the best performance with 80.1%, 81.3%, and

Table 3: Personalized inference performance (accuracy) for separately written characters.

| | Using Pen-Tail Coordinates | | | Using Reconstructed Handwriting | | |
|-------------------|----------------------------|-------|--------------|---------------------------------|-------|--------------|
| | SVM | KNN | Transformer | SVM | KNN | Transformer |
| Uppercase Letters | 0.712 | 0.697 | 0.803 | 0.733 | 0.739 | 0.842 |
| Lowercase Letters | 0.655 | 0.638 | 0.754 | 0.679 | 0.672 | 0.781 |
| Arabic Numerals | 0.791 | 0.787 | 0.895 | 0.812 | 0.799 | 0.913 |
| Specific Symbols | 0.745 | 0.741 | 0.852 | 0.758 | 0.762 | 0.875 |

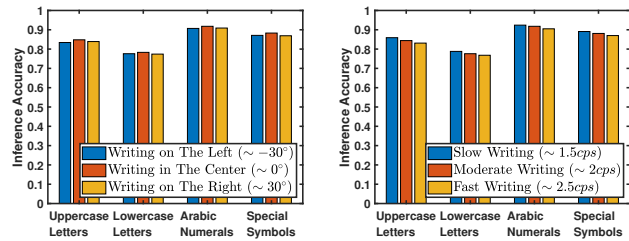


Figure 11: Impact of camera's viewing direction.

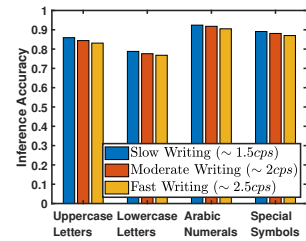


Figure 12: Impact of writing speed.

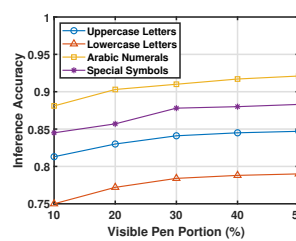


Figure 13: Impact of visible pen portion.

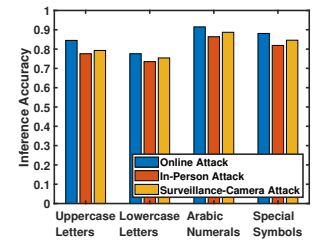


Figure 14: Online vs. in-person vs. surv.-cam attack.

79.7% accuracy, while the inference accuracy for letters “c”, “l”, and “o” degrades to 75.7%, 74.2%, and 75.1%, respectively.

We also explore how HW-Spy works for each participant to evaluate the impact of different people's writing styles. As shown in Fig. 10, HW-Spy's inference accuracy for the uppercase letters for most participants is above 83.8%, while the lowest inference accuracy is 82.4%. The inference accuracy for the lowercase letters achieved for most participants is above 76.9%, while the lowest inference accuracy is 75.4%. HW-Spy is shown to work well in general for different participants by handling different writing styles.

6.2.2 Consecutively Written Characters. We use the above transformer model trained with the handwriting reconstructed from the separately written characters to infer the consecutively written characters, where the characters can be disconnected or connected. The reconstructed handwriting from the consecutively written characters is used as the input to the transformer model. We also include SVM and KNN for comparison.

Disconnected Characters. Table 4 shows the performance of inferring the consecutive writing with disconnected characters. One can observe that SVM and KNN achieve 76.1% and 76.8% inference accuracies at the character level. The transformer-based model performs better with 87.9% inference accuracy. Furthermore, after applying the spell- and grammar-checkers, the word- and sentence-level inference accuracies achieved by the transformer-based model are increased to 90.3% and 92.0%, respectively.

Connected Characters. Table 4 also presents the inference performance of the consecutive writing with connected characters. SVM, KNN, and Transformer are observed to achieve a character-level inference accuracy of 60.3%, 60.9%, and 68.5%, respectively. The word- and sentence-level inference accuracies achieved by the transformer-based model are enhanced further to 70.7% and 71.8%, after applying the spell- and grammar-checkers. The inference performance for the connected characters is lower than that for the disconnected ones. However, when people fill in important forms, they usually avoid writing connected characters, which may cause

Table 4: Personalized inference performance (accuracy) for consecutively written characters.

| | Disconnected Characters | | | Connected Characters | | |
|-----------------|-------------------------|-------|--------------|----------------------|-------|--------------|
| | SVM | KNN | Transformer | SVM | KNN | Transformer |
| Character-level | 0.761 | 0.768 | 0.879 | 0.603 | 0.609 | 0.685 |
| Word-level | 0.794 | 0.815 | 0.903 | 0.612 | 0.623 | 0.707 |
| Sentence-level | 0.830 | 0.799 | 0.920 | 0.635 | 0.627 | 0.718 |

ambiguities and thus result in inconvenience and even incorrect interpretation of the user's inputted information.

6.3 Impact Factors

6.3.1 Camera's Viewing Direction. The victim may hand-write things at random positions in front of the screen, leading to different webcam's viewing directions, which may affect the performance of handwriting inference. So, we evaluate the effect of the webcam's viewing direction by comparing HW-Spy's performance in inferring the handwriting performed at the left (~ -30°), center (~ 0°), and right (~ 30°) parts of the scene. In particular, since the characters are written line-by-line during our data collection, the pen tail moves from left to right repeatedly. We thus split the written characters into three groups (i.e., writing on the left (~ -30°), writing at the center (~ 0°), and writing on the right (~ 30°)) based on the viewing directions of the webcam to their written positions and evaluate the handwriting inference performance for these three groups on separately written characters. As shown in Fig. 11, for inferring the uppercase letters, lowercase letters, Arabic numerals, and special symbols, our transformer-based model achieves similar performance when such characters are written on the left, at the center, or on the right of the webcam. No obvious differences have been observed, except that the handwriting done at the center of the camera is inferred with a slightly high accuracy for all types of characters. The results indicate that the impact of the camera's viewing direction is negligible and HW-Spy generally works well for different viewing directions.

6.3.2 Writing Speed. Different people write at different speeds, which could result in different characteristics for the written characters and thus impact the handwriting inference performance of HW-Spy. So, we evaluate HW-Spy's performance in inferring the handwriting performed at different speeds, i.e., characters per second (cps). Specifically, we divided the participants into three groups with slow writing (~ 1.5cps), moderate writing (~ 2cps), and fast writing (~ 2.5cps), and evaluate the handwriting inference performance for the three groups for writing the characters separately. As shown in Fig. 12, HW-Spy performs the best in inferring the characters written at slow speed, achieving an inference accuracy of 85.9%, 78.8%, 92.4%, and 89.1% for the uppercase letters, lowercase letters, Arabic numerals, and special symbols, respectively. As the writing speed increases, HW-Spy's handwriting inference performance does degrade slightly. Specifically, the inference accuracy for moderate writing speed decreases by up to 1.5% and that for fast writing decreases by up to 2.8%, compared to that for slow writing. These

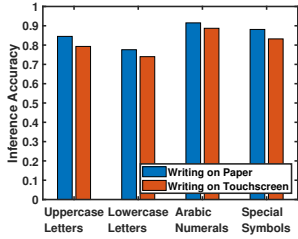


Figure 15: Writing on paper vs. on touchscreen.

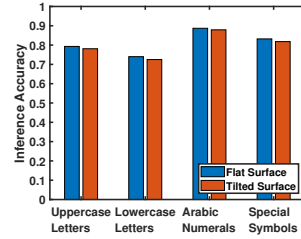


Figure 16: Flat vs. tilted writing surface.

results confirm that HW-Spy can handle different writing speeds and successfully infer the handwritten content.

6.3.3 Visible Pen Portion. The pen portion visible to the camera could affect the extraction of pen-tail coordinates and therefore influence the performance of handwriting inference. Owing to different camera placement, writing position, the pen’s length, and grip position, the visible pen portion varies across the writing process and is thus hard to evaluate. We, therefore, simulate varying levels of pen visibility by using different percentages of the segmented pen object for extracting the pen-tail coordinates. We then evaluate how this affects HW-Spy’s performance. As one can observe in Fig. 13, when only 10% of the pen is visible, HW-Spy achieves inference accuracies of 81.3%, 75.0%, 88.1%, and 84.5% for the uppercase letters, lowercase letters, Arabic numerals, and special symbols, respectively. When the visible pen portion is increased to 20% and 30%, the inference accuracies for these four types of characters are increased by an average of 1.8% and 3.1%, respectively. However, increasing the visible pen portion above 30% only makes slight impact on the inference performance.

6.4 Online vs. In-Person vs. Surveillance-Camera Attack

We also evaluate the *In-Person Attack* in which the victim is writing in a public space and the attacker video-records the victim’s pen-tail movements using a far-away camera. In particular, we ask 5 of the 20 participants to conduct the same experiments of Sec. 5.1.1 on paper in an office environment, during which the pen-tail movements were recorded using a smartphone’s rear camera placed around 5m away from the victim. We further evaluate the *Surveillance-Camera Attack* in which the customers’ writing motions in a bank/dealership/realty office are recorded by the surveillance camera hacked/installed by the adversary. In particular, we ask the same 5 participants to conduct the same experiments of Sec. 5.1.1 in an office environment, during which the pen-tail movements were recorded using a camera placed about 3m above the ground (\approx the height of an office ceiling), diagonally behind and above the victim. The angle between the camera and the victim ranges from 25° to 35° . For personalized handwriting inference, the adversary might obtain the training data from the same victim when the victim writes non-sensitive content without covering the writing surface. The written content can thus be captured by a surveillance camera and used for training. Another possibility is that the victim could write on some draft papers with unimportant/incomplete information and then throw them away or leave

them behind, which can be picked up by the adversary as the ground truth for the recorded pen-tail movements during training.

Fig. 14 compares the personalized handwriting inference performances of *In-Person*, *Online*, and *Surveillance-Camera* attacks. We can observe that for inferring all types of characters, *Surveillance-Camera Attack* performs slightly worse than *Online Attack*, achieving 79.3%, 75.4%, 88.7%, and 84.6% inference accuracies for uppercase letters, lowercase letters, Arabic numerals, and special symbols, respectively. In contrast, the inference accuracies for these four types of characters achieved by *In-Person Attack* further drops to 77.6%, 73.5%, 86.4%, and 81.9%, respectively. The reason for these is that the pen object segmented from the video frames captured by a far-away camera during *In-Person Attack*, or by a ceiling-mounted camera during *Surveillance-Camera Attack*, would have less pixels than those captured by the nearby (web)camera during *Online Attack*, which will make the derived pen-tail coordinates less accurate and thus affect the inference accuracy. Nonetheless, the results still show the feasibility of *In-Person Attack* and *Surveillance-Camera Attack*, which could be further enhanced by using more advanced devices like a camera with a telephoto lens or higher resolution.

6.5 Writing on Paper vs. on Touchscreen

The victim’s writing behavior on a touchscreen could be different from that on paper. For example, when writing on a touchscreen, the writing speed could be slower and the force on the pen could be lighter than writing on paper. This may cause the written characters to have different characteristics. We thus evaluate the impact of different writing mediums by comparing the personalized handwriting inference performance for the *Writing on Paper* and the *Writing on Touchscreen* scenarios. In particular, we ask the same 5 participants in Sec. 6.4 to conduct the same experiments of Sec. 5.1.1 on a touchscreen during an *Online Attack*. As shown in Fig. 15, HW-Spy performs slightly worse for the *Writing on Touchscreen* scenario, achieving accuracies of 79.3%, 74.0%, 88.7%, and 83.2% when inferring the upper- & lower-case letters, Arabic numerals, and special symbols, respectively. These results indicate that HW-Spy generally works well for different writing scenarios (mediums).

6.6 Flat vs. Tilted Writing Surface

While people typically write on flat surfaces, they may sometimes write on handheld or tilted surfaces, such as a touchscreen or a notepad on their lap, which may lead to different writing behaviors. We thus compare the personalized handwriting inference performance for the characters written on flat or tilted surfaces. In particular, we ask the same 5 participants in Sec. 6.4 to conduct the same

Table 5: Non-personalized handwriting reconstruction performance for separately written characters.

| | Uppercase | Lowercase | Arabic Numerals | Specific Numerals |
|-------------------|-----------|-----------|-----------------|-------------------|
| Avg. MSE (pixels) | 8.9 | 13.8 | 7.7 | 8.1 |

Table 6: Non-personalized handwriting reconstruction performance for consecutively written characters.

| | Disconnected Characters | Connected Characters |
|-------------------|-------------------------|----------------------|
| Avg. MSE (pixels) | 9.3 | 16.9 |

experiments of Sec. 5.1.1 on a touchscreen tilted at around 3° from horizontal during an *Online Attack*. We then compare HW-Spy’s performance on the newly collected data and the data collected in Sec. 6.5. The results presented in Fig. 16 shows that HW-Spy’s accuracy in inferring the characters written on the tilted surface decreases by an average of 1.2% compared to those written on the flat surface.

7 Non-Personalized Inference

We next evaluate the non-personalized performance of HW-Spy, where the victim’s data is excluded from training. Specifically, for each of the 20 participants to be selected as the target victim, we use the remaining 19 participants’ first-session data to train and validate the LSTM network and the transformer model. Training a non-personalized model for 50 epochs with a batch size of 32 and a learning rate of $1e^{-3}$ takes less than 180 minutes on four RTX-6000-Ada-48GB GPUs. The performances of non-personalized handwriting reconstruction and inference are then evaluated/tested using the target victim’s second-session data. The results presented below are averaged among all the participants.

7.1 Reconstruction of Non-Personalized Handwriting

Table 5 shows the non-personalized handwriting reconstruction performance for separately written characters. We can observe that the LSTM network achieves an average MSE of 8.9, 13.8, 7.7, and 8.1 pixels when reconstructing the uppercase letters, lowercase letters, Arabic numerals, and specific symbols, respectively. While for consecutively written characters, the LSTM network reconstructs the disconnected characters with an average MSE of 9.3 pixels and the connected characters with an average MSE of 16.9 pixels, as shown in Table 6. The non-personalized handwriting reconstruction still achieves a comparable performance to that of the personalized handwriting reconstruction presented in Sec. 6.1, even without the victim’s training data. The result indicates that the proposed LSTM network can be generally applied for handwriting reconstruction.

7.2 Inferring Non-personalized Handwriting

We present the performance of non-personalized handwriting inference for both separately and consecutively written characters.

Table 7: Non-personalized inference performance for separately written characters.

| | Uppercase | Lowercase | Arabic Numerals | Specific Symbols |
|----------|-----------|-----------|-----------------|------------------|
| Accuracy | 0.795 | 0.733 | 0.874 | 0.838 |

Table 8: Non-personalized inference performance (accuracy) for consecutively written characters.

| | Disconnected Characters | Connected Characters |
|-----------------|-------------------------|----------------------|
| Character-level | 0.845 | 0.667 |
| Word-level | 0.873 | 0.683 |
| Sentence-level | 0.898 | 0.701 |

7.2.1 Separately Written Characters. Table 7 shows the non-personalized inference performance for separately written characters. Our transformer-based model is shown to infer the uppercase letters, the lowercase letters, Arabic numerals, and the special symbols written separately by the victim with accuracies of 79.5%, 73.3%, 87.4%, and 83.8%, respectively, even without the target victim’s training data. This result corroborates the generality of our transformer model for handwriting inference.

7.2.2 Consecutively Written Characters. Table 8 shows the performance of inferring consecutively written characters without the writers’ information. For disconnected characters, our transformer-based model is shown to achieve a character-level accuracy of 84.5%. After applying the spell and grammar checkers, the word- and sentence-level accuracies are enhanced to 87.3% and 89.8%. For the connected characters, the inference accuracies at character-, word- and sentence-levels are 66.7%, 68.3%, and 70.1%, respectively.

7.3 Controlled vs. Freeform Writing

We also evaluate HW-Spy by asking the participants to write anything they want with no less than 100 words. For each of the 20 participants to be selected as the victim, we use the remaining 19 participants’ first-session data from the controlled experiments of Sec. 5.1.1 to train and validate the LSTM network and transformer-based model, which are then evaluated using the victim’s freeform-writing data. The results are averaged over all the participants. Specifically, HW-Spy reconstructs and infers freeform-writing with an average MSE of 9.7 pixels and an average character-level accuracy of 83.7%, which are close to its performance on controlled writing (9.3 pixels MSE and 84.5% character-level accuracy).

8 Ablation Study

We conduct an ablation study of the proposed LSTM network and transformer-based model. In particular, we fine-tune the models’ hyper-parameters with only the participants’ first-session data from the experiments of Sec. 5.1.1 to evaluate their impact on non-personalized handwriting reconstruction and classification for consecutively written (disconnected) characters. The results presented below are averaged among all the participants.

8.1 LSTM Network

8.1.1 Number of LSTM Layers. We tune the number of LSTM layers to evaluate its impact on the handwriting reconstruction. Specifically, with each layer having 400 hidden units, when there are 1, 2, 3, 4, and 5 LSTM layers in the network, the average MSE of handwriting reconstruction is 12.9, 12.4, 10.1, 10.8, and 11.3 pixels, respectively. Note stacking multiple LSTM layers can improve the learning process, but too many layers will lead to overfitting. So, we choose to use 3 LSTM layers for the handwriting reconstruction.

8.1.2 Number of Hidden Units. We also tune the number of hidden units within each LSTM layer to evaluate its impact on the handwriting reconstruction. In particular, for each of the three LSTM layers, we randomly set its hidden units to 200, 400, or 800, resulting in $3^3 = 27$ different combinations. When the numbers of hidden units for the three LSTM layers are set to 200/400/800, the

Table 9: Comparison with existing handwriting inference schemes w.r.t. character-level accuracy in %.

| | Sensor | Personalized Inference | | | | Non-Personalized Inference | | | |
|----------------------|--------------|------------------------|-------------------|-----------------|-----------------|----------------------------|-------------------|-----------------|-----------------|
| | | Uppercase Letters | Lowercase Letters | Arabic Numerals | Special Symbols | Uppercase Letters | Lowercase Letters | Arabic Numerals | Special Symbols |
| deWristified [45] | Acc/Gyro | 56.0 | 51.0 | - | - | 27.0 | 20.0 | - | - |
| WritingRecorder [51] | Mic | - | 77.3 | - | - | - | 65.0 | - | - |
| WritingHacker [52] | Mic/Acc | 64.9 | - | - | - | 26.8 | - | - | - |
| WordRecorder [12] | Mic | 81.0 | - | - | - | 74.8 | - | - | - |
| MagHacker [26] | Magnetometer | 80.1 | 77.7 | - | - | - | - | - | - |
| HackWrt [20] | Bluetooth | 74.0 | - | - | - | - | - | - | - |
| RadSee [56] | RF Sensor | - | - | - | - | Avg. 75.0 | | | |
| Patil et al. [32] | Camera | - | - | 61.7 | - | - | - | - | - |
| HW-Spy | Camera | 84.2 | 78.1 | 91.3 | 87.5 | 79.5 | 73.3 | 87.4 | 83.8 |

proposed network is observed to achieve the lowest average MSE of 9.0 pixels for the handwriting reconstruction.

8.2 Transformer-Based Model

8.2.1 Model Dimension. The model dimension d plays a crucial role in balancing the transformer model’s capacity, efficiency, and performance. So, we test the model with $d = 128, 256, 512, 1024$, and 2048. The model then achieves a character-level accuracy of 82.7%, 83.3%, 84.9%, 85.3%, and 85.8%, respectively. When $d > 512$, we observe only marginal performance improvements. Thus, to balance the performance and the computational cost, we set the model dimension d to 512.

8.2.2 Number of Attention Heads. The number of attention heads h in the transformer model also impacts the model’s expressiveness, computational efficiency, and memory usage. Given the model dimension $d = 512$ and the fact that h should be a divisor of d , we test the model with $h = 4, 8, 16$ and 32, respectively. The model then achieves a character-level accuracy of 81.8%, 84.9%, 85.2%, and 85.7%, respectively, in inferring the consecutively written (disconnected) characters. To balance the performance and the computational cost, we set the number of attention heads h to be 8.

8.2.3 Number of Transformer Encoders. The number of encoders N in the transformer model affects the model’s performance, computational cost, and ability to capture complex relationships in data. We thus test the model with $N = 1, 2, 3, 4$, and 5, respectively. The model then achieves a character-level accuracy of 78.8%, 81.6%, 84.9%, 84.1%, and 82.8%, respectively. This is because increasing the encoder layers can help capture richer patterns but too many layers can lead to overfitting. We thus set $N = 3$ for the proposed transformer model.

9 Comparison with State-of-the-Art

We compare the handwriting inference performance of HW-Spy with SOTA works mentioned in Sec. 2. As shown in Table 9, few of them evaluated both personalized and non-personalized performances.

For personalized performance where the models are trained with the victims’ handwriting data, WordRecorder [12] performs the best among the SOTA works by inferring the uppercase English alphabets with an accuracy of 81.0%, while it has not been evaluated for the lowercase English alphabets. MagHacker [26] achieves a

classification accuracy of 80.1% for the uppercase English alphabets and a classification accuracy of 78.1% for the lowercase English alphabets. DeWristfield [45] leveraging the smartwatch’s motion sensors for eavesdropping only achieves accuracies of 56.0% and 51.0% for inferring the uppercase and lowercase English alphabets, respectively. HackWrt [20] and WritingHacker [52] are only evaluated on uppercase English alphabets, achieving 74.0% and 64.9% inference accuracies. WritingRecorder [51] is only evaluated on lowercase English alphabets, achieving 77.3% inference accuracy. Patil *et al.* [32] evaluate their approach only on Arabic numerals, achieving a 61.7% accuracy in inferring single digits, which drastically outperforms their baseline of random guess. In contrast, HW-Spy outperforms all SOTA works in personalized performance, achieving accuracies of 84.2%, 78.1%, 91.3%, and 87.5% in inferring uppercase English alphabets, lowercase English alphabets, Arabic numerals, and special symbols, respectively.

For non-personalized performance where models are trained without the target victims’ handwriting data, RadSee [56] performs the best among the SOTA works by inferring the upper- and lowercase English alphabets and Arabic numerals with an average accuracy of 75.0%, but the detailed performance for upper/lowercase alphabets and Arabic numerals is missing. WordRecorder [12] achieves a 74.8% accuracy for inferring the uppercase English alphabets. WritingRecorder [51] achieves a 65.0% inference accuracy for the lowercase English alphabets. The inference accuracies achieved by the other two SOTA works are all below 30.0%. In particular, deWristfield [45] infers the upper- and lower-case English alphabets with 27% and 20% accuracies, respectively. WritingHacker [52] achieves a 26.8% inference accuracy. These results show a big gap between the personalized and non-personalized performances achieved by the SOTA works, which limits their applicability. In contrast, HW-Spy achieves a non-personalized inference performance for the uppercase English alphabets, lowercase English alphabets, Arabic numerals, and special symbols, with accuracies of 79.5%, 73.3%, 87.4%, and 83.8%, respectively, which is comparable to its personalized performance.

10 Discussion

10.1 Defenses

Since HW-Spy is shown to successfully infer the written content in realistic scenarios, those participating in online meetings, working

in public spaces, or writing under surveillance cameras, should take measures to protect their written content. In addition to checking nearby areas for suspicious cameras and other sensors, they should also consider keeping their pens out of the webcam's angle of view while writing during online meetings or using physical covers and/or their body parts to hide all pen/hand movements while writing in a public space or writing under surveillance cameras. This is likely the easiest and most effective defense against HW-Spy. Specifically those companies providing online meeting platforms may also consider adding extra blurring functions that can be customized by the users to hide the (sensitive) written content, similarly to the background blurring methods currently used by most online meeting platforms.

10.2 Limitations

Dependency on pen-tail tracking accuracy: HW-Spy leverages image segmentation to extract the pen object from the video frame's background and calculates the pen-tail coordinates based on the extracted pixel area. Therefore, the pen-tail tracking accuracy is greatly impacted by the image segmentation algorithm, which may not work well in certain circumstances. For example, if the pen's color is close to the color of the background (such as the user's clothes or furniture), the image segmentation may not extract the accurate pen object, thus causing errors in the calculation of pen-tail coordinates. To address this issue, we may need to tailor the image segmentation algorithm to the background.

Camera direction/distance/resolution: Although in Sec. 6.3.1 we have studied the impact of the camera's viewing direction on HW-Spy, we only considered the camera directions from a horizontal perspective. But the cameras can also be tilted during their real-life use, creating different vertical viewing directions, which may also impact the extraction of the pen-tail coordinates. In addition, the distance between the camera and the pen tail, as well as the camera's resolution, will both impact the resolution of the pen tail recorded in the video, which results in different numbers of pixels to be used for calculating the pen-tail coordinates. Therefore, we need a comprehensive study of the camera's direction, distance, and resolution. Moreover, a more advanced attacker can leverage stereo cameras, depth cameras, or even LiDARs to acquire 3D movements of the pen tail to infer the handwriting with a higher accuracy, which is also worth studying.

11 Conclusion

We have proposed HW-Spy, a video-based handwriting inference attack, which can infer the handwritten content from the victim's pen-tail movements captured by the webcam during an online meeting, by a distant smartphone/tablet camera in public spaces, or by the surveillance camera in a bank/dealership/realty office. In particular, we used image segmentation to extract the pen object from the recorded video and applied a linear regression to acquire the pen's skeleton, whose top point is then used as the pen-tail coordinates for the current video frame. We then identified the stroke-associated movements from the pen's in-air movements using a 1D U-Net model trained for stroke mask prediction and derived motion features to segment the characters. The segmented pen-tail coordinates associated with each character are fed into an

LSTM-based network for handwriting reconstruction. In the last step, the reconstructed handwriting coordinates of each character are processed by a transformer-based model, which leverages the attention mechanism to learn the characteristics of different written characters and output the predicted character labels. HW-Spy is shown to outperform SOTA works, achieving comparable performance for both personalized and non-personalized handwriting inferences, i.e., achieving accuracies of up to 84.2% and 79.5% for personalized non-personalized handwriting inference, respectively.

Acknowledgments

The work reported in this paper was supported in part by the US National Science Foundation under Grant No. 2245223.

References

- [1] Dosovitskiy Alexey. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [2] Kamran Ali, Alex X Liu, Wei Wang, and Muhammad Shahzad. 2015. Keystroke recognition using wifi signals. In *Proceedings of the 21st annual international conference on mobile computing and networking*, 90–102.
- [3] Christoph Amma, Marcus Georgi, and Tanja Schultz. 2012. Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3D-space handwriting with inertial sensors. In *2012 16th International Symposium on Wearable Computers*. IEEE, 52–59.
- [4] Dmitri Asonov and Rakesh Agrawal. 2004. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy*, 2004. *Proceedings*. IEEE, 3–11.
- [5] Davide Balzarotti, Marco Cova, and Giovanni Vigna. 2008. Clearshot: Eavesdropping on keyboard input from video. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 170–183.
- [6] John L Barron, David J Fleet, Steven S Beauchemin, and TA Burkitt. 1992. Performance of optical flow techniques. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 236–237.
- [7] B Basavaprasad and S Hegadi Ravindra. 2014. A survey on traditional and graph theoretical techniques for image segmentation. *Int. J. Comput. Appl* 975 (2014), 8887.
- [8] Hancheng Cao, Chia-Jung Lee, Shamsi Iqbal, Mary Czerwinski, Priscilla NY Wong, Sean Rintel, Brent Hecht, Jaime Teevan, and Longqi Yang. 2021. Large scale analysis of multitasking behavior during remote meetings. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.
- [9] Yimin Chen, Tao Li, Rui Zhang, Yanchao Zhang, and Terri Hedgpeth. 2018. Eyetell: Video-assisted touchscreen keystroke inference from eye movements. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 144–160.
- [10] Paula Delgado-Santos, Ruben Tolosana, Richard Guest, Farzin Deravi, and Ruben Vera-Rodriguez. 2022. Exploring Transformers for Behavioural Biometrics: A Case Study in Gait Recognition. *arXiv preprint arXiv:2206.01441* (2022).
- [11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xi-aohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [12] Haishi Du, Ping Li, Hao Zhou, Wei Gong, Gan Luo, and Panlong Yang. 2018. Wordrecorder: Accurate acoustic-based handwriting recognition using deep learning. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1448–1456.
- [13] Song Fang, Ian Markwood, Yao Liu, Shangqing Zhao, Zhuo Lu, and Haojin Zhu. 2018. No training hurdles: Fast training-agnostic attacks to infer your typing. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1747–1760.
- [14] Fake Person Generator. [n. d.]. Random Address Generator. <https://www.fakepersongenerator.com/random-address> [Accessed: 2024-08-01].
- [15] Swarnendu Ghosh, Nibaran Das, Ishita Das, and Ujjwal Maulik. 2019. Understanding deep learning techniques for image segmentation. *ACM computing surveys (CSUR)* 52, 4 (2019), 1–35.
- [16] Grammarly Inc. [n. d.]. Grammarly. <https://www.grammarly.com/> [Accessed: 2024-10-05].
- [17] Berthold KP Horn and Brian G Schunck. 1981. Determining optical flow. *Artificial intelligence* 17, 1-3 (1981), 185–203.
- [18] Jingyang Hu, Hongbo Wang, Tianyue Zheng, Jingzhi Hu, Zhe Chen, Hongbo Jiang, and Jun Luo. 2023. Password-Stealing without Hacking: Wi-Fi Enabled Practical Keystroke Eavesdropping. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 239–252.

- [19] Wenqiang Jin, Srinivasan Murali, Huadi Zhu, and Ming Li. 2021. Periscope: A keystroke inference attack using human coupled electromagnetic emanations. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 700–714.
- [20] Aaron Kinfe, Chijung Jung, Kai Lin, Marshall Clyburn, and Fnu Suya. 2023. HackWrt: Network Traffic-Based Eavesdropping of Handwriting. In *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. 55–60.
- [21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4015–4026.
- [22] Frederic F Leymarie and Benjamin B Kimia. 2008. From the infinitely large to the infinitely small: Applications of medial symmetry representations of shape. *Medial representations: mathematics, algorithms and applications* (2008), 327–351.
- [23] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na Ruan. 2016. When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 1068–1079.
- [24] John Lim, True Price, Fabian Monrose, and Jan-Michael Frahm. 2020. Revisiting the threat space for vision-based keystroke inference attacks. In *Computer Vision—ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part V*. 16. Springer, 449–461.
- [25] Xiangyu Liu, Zhe Zhou, Wenrui Diao, Zhou Li, and Kehuan Zhang. 2015. When good becomes evil: Keystroke inference with smartwatch. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1273–1285.
- [26] Yihao Liu, Kai Huang, Xingzhe Song, Boyuan Yang, and Wei Gao. 2020. Maghacker: eavesdropping on stylus pen writing via magnetic sensing from commodity mobile devices. In *Proceedings of the 18th international conference on mobile systems, applications, and services*. 148–160.
- [27] Anindya Maiti, Murtuza Jadhwal, Jibo He, and Igor Bilogrevic. 2018. Side-channel inference attacks on mobile keypads using smartwatches. *IEEE Transactions on Mobile Computing* 17, 9 (2018), 2180–2194.
- [28] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. 2011. (sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*. 551–562.
- [29] Pierre-François Marteau. 2008. Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence* 31, 2 (2008), 306–318.
- [30] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. 2021. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3523–3542.
- [31] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. Fingerio: Using active sonar for fine-grained finger tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1515–1525.
- [32] Neil Patil, Brian Cui, and Hovav Shacham. 2019. Big Wave Shoulder Surfing. <https://neilpatil.me/doc/surfing.pdf>. Accessed July 10, 2025.
- [33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.
- [34] Rahul Raguram, Andrew M White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. 2011. iSpy: automatic reconstruction of typed input from compromising reflections. In *Proceedings of the 18th ACM conference on Computer and communications security*. 527–536.
- [35] RANDOM.ORG. [n. d.]. Random Password Generator. <https://www.random.org/passwords/> [Accessed: 2024-08-01].
- [36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III* 18. Springer, 234–241.
- [37] Mohd Sabra, Anindya Maiti, and Murtuza Jadhwal. 2020. Zoom on the keystrokes: Exploiting video calls for keystroke inference attacks. *arXiv preprint arXiv:2010.12078* (2020).
- [38] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir V Poha. 2014. Beware, your hands reveal your secrets!. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 904–917.
- [39] Giuseppe Stragapede, Paula Delgado-Santos, Ruben Tolosana, Ruben Vera-Rodriguez, Richard Guest, and Aythami Morales. 2023. Mobile keystroke biometrics using transformers. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*. IEEE, 1–6.
- [40] David Temoshok, James Fenton, Yee-Yin Choong, Naomi Lefkowitz, Andrew Regenscheid, and Justin Richer. 2024. *Digital Identity Guidelines: Authentication and Authenticator Management*. Technical Report. National Institute of Standards and Technology.
- [41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [42] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. 2016. Friend or foe? Your wearable devices reveal your personal pin. In *Proceedings of the 11th ACM on Asia conference on computer and communications security*. 189–200.
- [43] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the 21st annual international conference on mobile computing and networking*. 155–166.
- [44] Yao Wang, Wandong Cai, Tao Gu, and Wei Shao. 2019. Your eyes reveal your secrets: An eye movement based password inference on smartphone. *IEEE transactions on mobile computing* 19, 11 (2019), 2714–2730.
- [45] Raveen Wijewickrama, Anindya Maiti, and Murtuza Jadhwal. 2019. deWristified: handwriting inference using wrist-based motion sensors revisited. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. 49–59.
- [46] Qingxin Xia, Feng Hong, Yuan Feng, and Zhongwen Guo. 2018. Motionhacker: Motion sensor based eavesdropping on handwriting via smartwatch. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPs)*. IEEE, 468–473.
- [47] Chao Xu, Parth H Pathak, and Prasant Mohapatra. 2015. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th international workshop on mobile computing systems and applications*. 9–14.
- [48] Yi Xu, Jared Heinly, Andrew M White, Fabian Monrose, and Jan-Michael Frahm. 2013. Seeing double: Reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 1063–1074.
- [49] Edwin Yang, Qiuye He, and Song Fang. 2022. WINK: Wireless Inference of Numerical Keystrokes via Zero-Training Spatiotemporal Analysis. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 3033–3047.
- [50] Zhuolin Yang, Yuxin Chen, Zain Sarwar, Hadleigh Schwartz, Ben Y Zhao, and Haitao Zhang. 2023. Towards a general video-based keystroke inference attack. In *32nd USENIX Security Symposium (USENIX Security 23)*. 141–158.
- [51] Huanpu Yin, Anfu Zhou, Guangyuan Su, Bo Chen, Liang Liu, and Huadong Ma. 2020. Learning to recognize handwriting input with acoustic features. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–26.
- [52] Tuo Yu, Haiming Jin, and Klara Nahrstedt. 2016. Writinghacker: audio based eavesdropping of handwriting via mobile devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 463–473.
- [53] Qinggang Yue, Zhen Ling, Xinwen Fu, Benyuan Liu, Kui Ren, and Wei Zhao. 2014. Blind recognition of touched keys on mobile devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1403–1414.
- [54] Qinggang Yue, Zhen Ling, Wei Yu, Benyuan Liu, and Xinwen Fu. 2015. Blind recognition of text input on mobile devices via natural language processing. In *Proceedings of the 2015 Workshop on Privacy-Aware Mobile Computing*. 19–24.
- [55] Chen Yunfang, Zhu Yihong, Zhou Hao, Chen Wei, and Zhang Wei. 2018. Enhanced keystroke recognition based on moving distance of keystrokes through wifi. In *Network and System Security: 12th International Conference, NSS 2018, Hong Kong, China, August 27–29, 2018, Proceedings 12*. Springer, 237–250.
- [56] Shichen Zhang, Qijun Wang, Maolin Gan, Zhichao Cao, and Huacheng Zeng. 2025. RadSee: See Your Handwriting Through Walls Using FMCW Radar. *Network and Distributed System Security (NDSS) Symposium* (2025).
- [57] Yunting Zhang, Jiliang Wang, Weiyi Wang, Zhao Wang, and Yunhao Liu. 2018. Vermier: Accurate and fast acoustic motion tracking using mobile devices. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1709–1717.
- [58] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.
- [59] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. 2014. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 453–464.
- [60] Li Zhuang, Feng Zhou, and J Doug Tygar. 2009. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)* 13, 1 (2009), 1–26.