S^3: Side-Channel Attack on Stylus Pencil through Sensors

HABIBA FARRUKH, Purdue University, USA
TINGHAN YANG, Purdue University, USA
HANWEN XU, Tsinghua University, China
YUXUAN YIN, Tsinghua University, China
HE WANG, Purdue University, USA
Z. BERKAY CELIK, Purdue University, USA

With smart devices being an essential part of our everyday lives, unsupervised access to the mobile sensors' data can result in a multitude of side-channel attacks. In this paper, we study potential data leaks from Apple Pencil (2nd generation) supported by the Apple iPad Pro, the latest stylus pen which attaches to the iPad body magnetically for charging. We observe that the Pencil's body affects the magnetic readings sensed by the iPad's magnetometer when a user is using the Pencil. Therefore, we ask: *Can we infer what a user is writing on the iPad screen with the Apple Pencil, given access to only the iPad's motion sensors' data?* To answer this question, we present **S**ide-channel attack on **S**tylus pencil through **S**ensors (S^3), a system that identifies what a user is writing from motion sensor readings. We first use the sharp fluctuations in the motion sensors' data to determine when a user is writing on the iPad. We then introduce a high-dimensional particle filter to track the location and orientation of the Pencil during usage. Lastly, to guide particles, we build the Pencil's magnetic map serving as a bridge between the measured magnetic data and the Pencil location and orientation. We evaluate S^3 with 10 subjects and demonstrate that we correctly identify 93.9%, 96%, 97.9%, and 93.33% of the letters, numbers, shapes, and words by only having access to the motion sensors' data.

CCS Concepts: • **Security and privacy** → **Access control**.

Additional Key Words and Phrases: side-channel attack, user privacy, stylus pencils

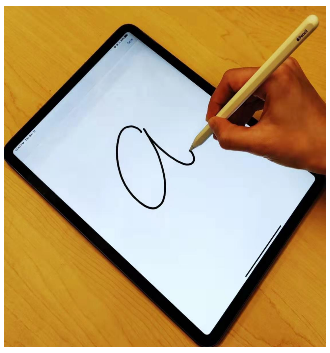
ACM Reference Format:
Habiba Farrukh, Tinghan Yang, Hanwen Xu, Yuxuan Yin, He Wang, and Z. Berkay Celik. 2021. S^3: Side-Channel Attack on Stylus Pencil through Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 1, Article 8 (March 2021), 25 pages. https://doi.org/10.1145/3448085

1 INTRODUCTION

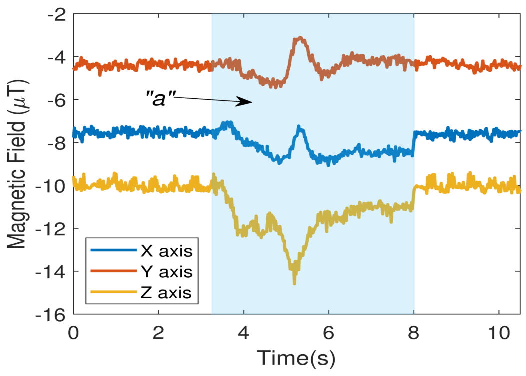
Modern-day smart devices come embedded with various sensors, enabling a vast range of applications in activity recognition, context awareness, mobile health, and productivity. Unfortunately, these sensors are also gateways for unintended information leakage about users' activities. Unauthorized access to users' personal information, habits, behaviors, and preferences through sensor data has long been a major security and privacy concern.

Authors' addresses: Habiba Farrukh, hfarrukh@purdue.edu, Purdue University, 305 N. University St, West Lafayette, IN, USA, 47907; Tinghan Yang, yang1683@mpurdue.edu, Purdue University, 305 N. University St, West Lafayette, IN, USA, 47907; Hanwen Xu, xuhw20@mails.tsinghua.edu.cn, Tsinghua University, Beijing, China; Yuxuan Yin, yin-yx16@tsinghua.org.cn, Tsinghua University, Beijing, China; He Wang, hw@purdue.edu, Purdue University, 305 N. University St, West Lafayette, IN, USA, 47907; Z. Berkay Celik, zcelik@purdue.edu, Purdue University, 305 N. University St, West Lafayette, IN, USA, 47907.

2474-9567/2021/3-ART8 $15.00
https://doi.org/10.1145/3448085



(a)



(b)

Fig. 1. Example of magnetometer readings when a user writes on the iPad using Apple Pencil.

Mobile operating systems such as Android and iOS require users' explicit permission to grant access to sensitive data (e.g., GPS locations, microphone, and camera recordings) to an application. However, data from more generic sensors such as magnetometer, accelerometer and gyroscope are less regulated, and often can be accessed by applications without users' permission. Unmonitored access to these sensors' data has recently opened the door to a multitude of side-channel attacks. Prior efforts have unveiled security and privacy concerns that result from applications having access to motion sensors' data. Such works propose attacks targeted at inferring users' privacy-sensitive data such as touch actions and keystrokes [1–3], passwords and PIN codes [2, 4], and application and webpage fingerprinting [5, 6]. Unfortunately, side-channel attacks continue to be a major source of concern to users and developers due to frequent software updates and new devices introduced in the market.

With the ever-increasing popularity of large-screen handheld devices, many manufacturers have started equipping their products with *stylus pencils* to improve user experience. In this paper, we study the recently launched "Apple Pencil" used with one of Apple's most popular devices, the iPad Pro. The Apple Pencil lets users write, draw, and make selections in a variety of generic and custom-designed applications with ease. The second generation of this product launched in 2018 offers a convenient new feature where the pencil pairs with the iPad to charge wirelessly. The pencil's body is embedded with multiple magnets, allowing it to magnetically attach to the iPad's body. In this paper, we present a novel side-channel attack on the Apple Pencil. The magnetic readings sensed by the iPad's magnetometer are impacted when the user interacts with the screen using the pencil. We show that, given that the Apple Pencil is often used to fill text fields in applications [7], an adversary can infer a victim's private data such as passwords, notes, text messages and signatures by only tracking the movement of the pencil through the motion sensors' data.

To illustrate our idea, Figure 2b presents the recorded magnetometer data in X, Y, and Z directions when a user writes the character 'a' on the iPad with the Pencil, as shown in Figure 1a. We show that an adversary can infer what users are writing on the iPad's screen by observing the fluctuations in the magnetic readings by *only* having access to motion sensors' data. We note that the attack does not rely on any touch information (i.e., the pencil tip position) since iOS does not allow third-party applications running in the background to access this information.

As observed in Figure 1, we can readily determine when the pencil is used to write on the screen from the magnetic data. However, to infer the *contents* written on the screen, the magnetic readings require detailed analysis. In our preliminary experiments, we observe various subtle fluctuations in magnetic readings characterized by different letters and words written on the screen. To illustrate, we present the recorded magnetometer data

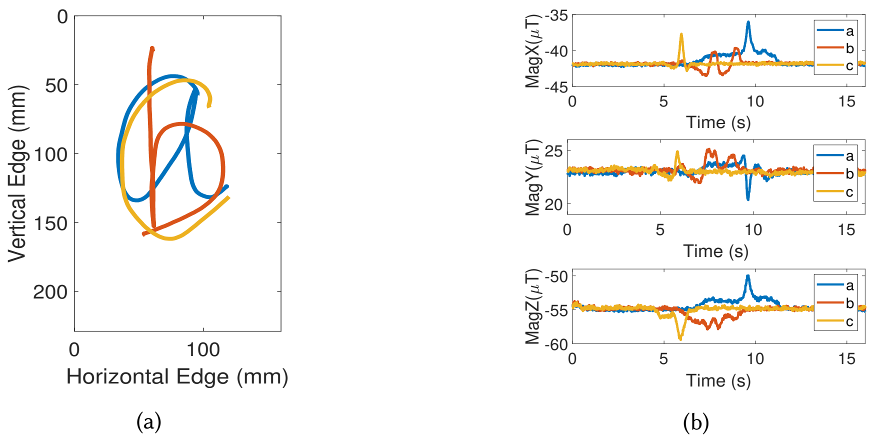


Fig. 2. The magnetometer readings change when a user writes different characters on the screen.

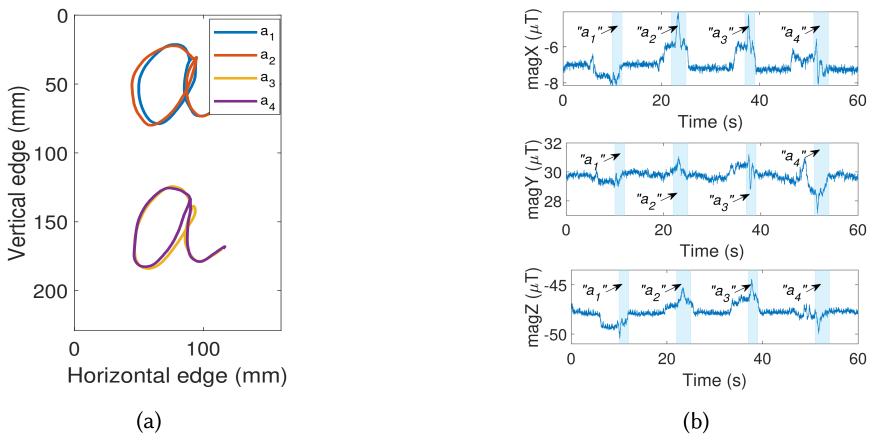


Fig. 3. The magnetometer readings change when a user writes the same character on the screen.

in Figure 2 when the letters 'a', 'b' and 'c' are written on the iPad's screen (Figure 2a). Figure 2b shows the magnetometer readings in three axes, where fluctuations in the magnetometer readings are different for each of the three characters. One intuitive approach to extract information from these readings is to train a learning model. This model can then be used to identify useful patterns (e.g., letters, numbers, shapes) from similarities between the magnetic data over time for a given set of characters. Yet, this approach works when the pencil's magnetic impact is consistent when it is held at different locations or with different orientations on the screen for a given character. To determine whether this is the case for Apple Pencil and iPad Pro, we wrote the character 'a' with the pencil at different locations and orientations on the iPad's screen. Figure 3a illustrates two letters written at different locations (a_1 and a_3) and two letters in the same location when the pencil is rolled by 180° (a_3 and a_4). We observe that magnetometer readings vary significantly, although the traces of characters are visually the same.

Based on these observations, we design a tracking algorithm to track the pencil's tip movement using the magnetic field data to identify users' writing. Building such an algorithm requires finding the correct mapping between a given magnetic reading and the pencil tip's location and its orientation with respect to the screen. A common method for finding such mappings is through war-driving. Unfortunately, the mapping space is five-dimensional since we want to track the pencil's 2D location on the screen *while* keeping a record of its 3D orientation. War-driving for all possible locations and orientations of the pencil on the iPad leads to a huge search space and necessitates a huge amount of human effort.

To address this, we reduce the war-driving space by building a 3D magnetic field map *around the pencil* instead of determining the magnetic field for the screen itself. We collect magnetometer data while writing with different orientations of the pencil at different locations on the screen. We employ a computer vision-based approach for pose-estimation to track the pencil's 3D orientation since iOS's touch API does not provide full information about the pencil's 3D orientation. We apply Kalman filtering [8] and smoothing to the estimated pencil orientations to remove noisy data. We then use this orientation along with the pencil tip location and magnetic data to build the magnetic map with respect to the pencil. Lastly, to further improve our magnetic field map precision, we adopt a magnetic vector field reconstruction technique [9], which uses the divergence and curl properties [10] of the magnetic fields for optimizing the reconstruction process. This map generation is conducted offline and does not require information collected from the target user.

Using the designed magnetic field map, we build a multi-dimensional *particle filter* [11] to track the status of the pencil on the iPad's screen, which solely operates with the data collected from motion sensors. The particles' state includes the pencil tip's location and the 3D orientation of the pencil represented as quaternion vectors [12]. We use the human writing behavior to guide the particles' initialization and transition (components of our particle filter). Additionally, we used KLD resampling [13] to improve efficiency and incorporated particles' history to handle scattered particle clusters that may exist at the beginning of the tracking process. For an end-to-end attack, we also analyzed the variance in the magnetometer, accelerometer, and gyroscope readings to estimate when the user begins and ends writing on the screen.

We implement our system, S^3 (**S**ide-channel attack on **S**tylus pencil through **S**ensors), on Apple 11" iPad Pro running iOS 12.0. We evaluate S^3 with 10 subjects, where the subjects write different letters, numbers, shapes, and words at different locations of the screen. We show that an adversary (randomly chosen from the subjects) is able to correctly identify 93.9%, 96%, and 97.9% of the written letters, numbers, and shapes, and English words with an accuracy of 93.33%.

The main contributions of this paper are summarized as follows:

- We unveil a privacy leakage through motion sensor data in modern tablet devices due to the introduction of stylus pencils with embedded magnets.
- We design a novel tracking algorithm by constructing a magnetic map of the pencil and building a sophisticated multi-dimensional particle filter that can track the user's writing.
- We implemented our system on an iPad Pro with Apple Pencil. A thorough evaluation with 10 human subjects demonstrates its high accuracy in uncovering letters, numbers, shapes, and words in a variety of scenarios, without significant overhead.

2 THREAT MODEL

We use Apple's iPad and Pencil (2nd generation) to demonstrate a proof-of-concept of inferring information about what the user is writing from motion sensors data. However, our attack can be a threat to any mobile device with a stylus pen support using embedded magnets. The sensitive information being leaked out from user writings can be an unprecedented threat to the confidentiality of user data processed by the writing apps; an adversary can infer passwords, text messages, secure access codes, and other secrets of the user.

We consider an adversary that controls a malicious native application installed on the victim's device, which has access to the motion sensors data and runs in the background during data collection. The native application does not have any information about other running applications and does not have access to system resources. To detail, the iOS applications, by default, have access to the motion sensor data, and only web applications from iOS 12.2 and onwards require explicit user permission. Starting from iOS 5, an application can stay active in the background if it performs specific tasks such as playing audio, receiving updates from a server, and using location services. Given these facts, an adversary can easily mimic a benign legitimate app such as a fitness and activity tracker to stay active in the background to collect motion data. The application periodically logs the motion sensors' data, including accelerometer, gyroscope, and magnetometer readings. The recorded sensor data is stored locally and sent to a remote server for processing. These processes incur minimal energy and computation overhead in order to remain undetected (Detailed in Section 6).

We additionally assume that the adversary can obtain or learn a model to capture the users' writing behavior, which is used to predict how the Pencil's movement changes over time through Pencil's previous positions. The adversary could learn such a model by collecting handwriting samples (various letters, numbers, shapes, etc.) from a small number of people (up to 3 users is sufficient for the attack to succeed as detailed in our evaluation in Section 6). To build this model, an adversary uses a computer vision-based technique (See Section 5.1) to track the Pencil's orientation while logging the Pencil's location and motion sensors' data. We note that the handwriting samples *are not collected from the potential attack victims* and are solely obtained from the adversary and their accomplices. After the model is learned, the adversary uses the model to infer the free-form handwriting of unaware users *solely by collecting motion sensor data*. We emphasize that the Pencil location and orientation data are collected only during the training process. An adversary does not need to access this information at attack time to infer user writings.

3 PENCIL TRACKING: A FIRST LOOK

In our quest for inferring the user's writing from the sensors' data, we begin by exploring the simplest case: *assume that the user always holds the Pencil in a fixed orientation, i.e., vertically such that the Pencil altitude remains* $90°$. In this case, to track how the Pencil moves on the screen, we first determine a mapping between the magnetic readings and the location of the Pencil tip. We define this mapping as the *2D magnetic map*.

3.1 2D Magnetic Map

The 2D magnetic field map would be a function, f, that returns the magnetic readings corresponding to a given location on the screen i.e.

$$(m_x, m_y, m_z) = f(x, y) \tag{1}$$

where x and y are the coordinates of the Pencil tip location and m_x, m_y and m_z are the magnetic reading in x, y and z directions. We consider the shorter edge of the iPad as x-axis and the longer edge as y-axis.

To build this map, we draw on the screen using the Pencil such that we cover the entire screen. We use a custom drawing application that logs the x and y coordinates of the Pencil on the screen using iOS touch API, at a rate of 120Hz. The magnetic data is recorded in the background at a frequency of 50Hz. The timestamps recorded with the magnetic and touch data are used to align the magnetic readings with each of the touch samples. We start collecting the magnetic data a few minutes before drawing (when the Pencil is detached from the iPad and is not on the screen) to sense the ambient magnetic field. The average of the ambient magnetic data is subtracted from the collected data to eliminate the magnetic impact of the surroundings. Figure 4 shows the 2D magnetic field map in the x, y and z directions.

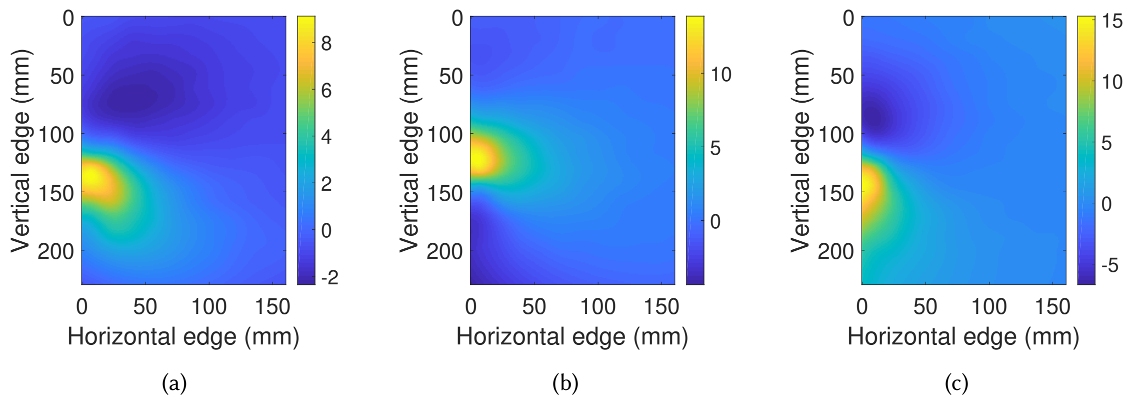


Fig. 4. 2D magnetic map: the magnetic impact of the Pencil at different locations on the screen in (a) X, (b) Y, and (c) Z dimensions.

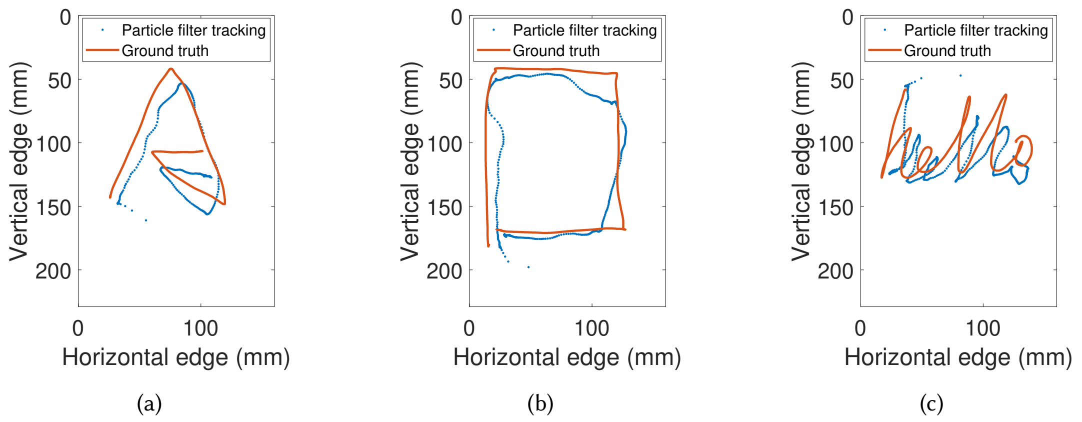


Fig. 5. Tracking results from a simple implementation of particle filter.

3.2 2D Tracking

Once we have built the 2D magnetic map, we ask the question: *how can we track the Pencil location using only the magnetic data?* Here we borrow a standard tracking algorithm, *particle filter*, to track the Pencil's location. Particle filtering is a probabilistic approximation algorithm for state estimation problems [11]. In particular, for object location estimation, particle filter operates by maintaining a probability distribution for the location estimate at time t, which is represented by a set of weighted samples called particles. The state of these particles is updated based on a motion model for the object, and the weights for these particles are assigned at each timestamp by a likelihood model for the recorded sensor data. The particles are then resampled at each timestamp using importance sampling to choose a new set of particles according to the weights of the prior samples.

We define the state vector for the particle filter as s_t, representing the Pencil tip location. Here, t represents the timestamp, which is updated at a frequency of 50Hz. For the simple case, we define the Pencil movement model as:

$$s_{t+1} = s_t + b_t \tag{2}$$

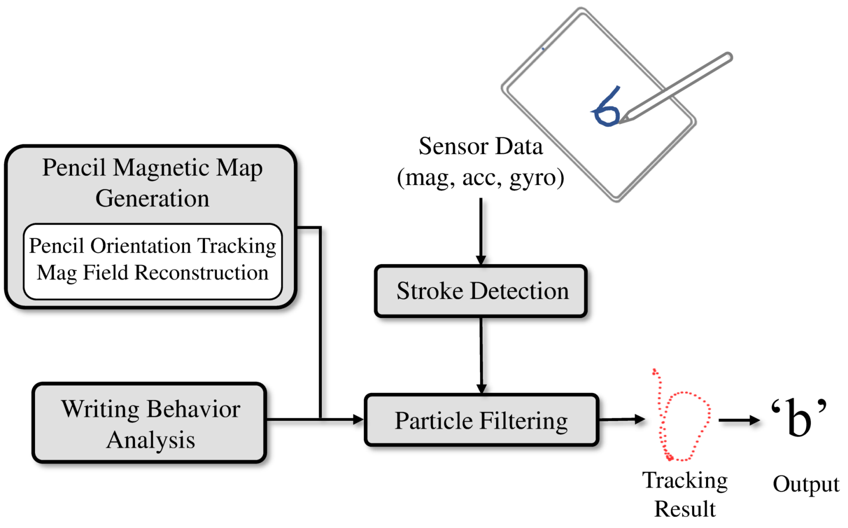

Pencil Magnetic Map Generation
Pencil Orientation Tracking
Mag Field Reconstruction
Writing Behavior Analysis
Sensor Data (mag, acc, gyro)
Stroke Detection
Particle Filtering
Tracking Result
'b'
Output

Fig. 6. System architecture.

where b_t is a random perturbation added to the state, with a Gaussian distribution.

Weights are assigned to each particle at each timestamp with the function:

$$w_t^i = \exp\left(\frac{(m_t - r_t)^2}{\sigma}\right) \tag{3}$$

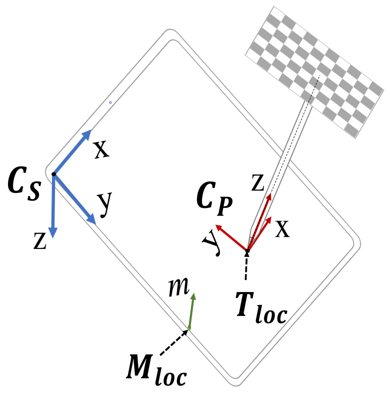
Here r_t is the magnetic readings obtained from the magnetometer at time t and m_t is the queried magnetic readings given state is s_t, using Equation 1. The location of the Pencil tip at each timestamp is the weighted average of the particles.

Figure 5 shows the results of this simple implementation of particle filter when used to track a letter (*'A'* - Figure 5a), a shape (a square - Figure 5b) and a word (*'hello'* - Figure 5c). Even though we assumed that the Pencil orientation is fixed for this case, the tracking results show promise that we can infer what the user is writing based on sensors' data.

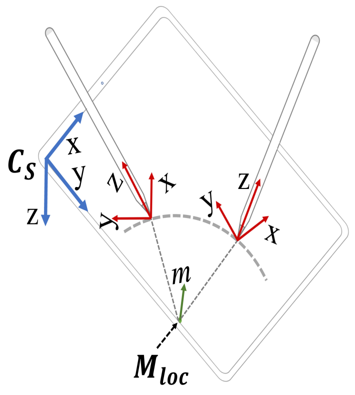
4 SYSTEM ARCHITECTURE

This section presents the functional overview of our system S^3as shown in Figure 6. We will detail each component of our system in Section 5.

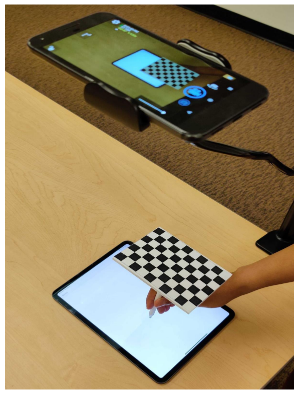
To launch a realistic attack, our system should find out what the user is writing in their natural handwriting style, unlike the fixed orientation case in the previous section. For this purpose, we need a mapping between the magnetic data collected and the status of the Pencil, which now includes location and orientation, i.e., 5 degrees of freedom ($\langle x, y\rangle$ location and 3D orientation of the Pencil). We generate this map through war-driving and reduce the effort involved by building the magnetic map around the Pencil. We extend computer vision techniques to track the Pencil's orientation by attaching a checkerboard on top of the Pencil and recording the Pencil movement with a camera. A magnetic field reconstruction technique [9] is used to remove noise from the collected magnetic data and generate a continuous magnetic field map for the Pencil. This magnetic map for a given iPad and Pencil can be built solely by the attacker. It is a one-time effort since the same map is used to track the user's handwriting in different locations and environments.



(a)



(b)



(c)

Fig. 7. (a) Screen and Pencil coordinate systems and the location of the magnetometer. (b) An example that the location of the magnetometer in the Pencil coordinate system is the same, even when the Pencil's locations and orientations are different. (c) Setup for tracking the Pencil orientation.

We also build a model for writing behavior by writing different letters, numbers, and shapes at different locations on the screen multiple times while tracking the Pencil orientation with a camera, as previously mentioned. This model finds the relationship between orientation and location changes while writing via linear regression. It predicts how these parameters should change at a given timestamp with respect to the Pencil's previous position. Similar to the magnetic map, this writing behavior model is also trained on data collected from the attacker and their accomplices and does not involve the victim.

The motion sensors' data (including magnetometer, accelerometer, and gyroscope data) collected by the attacker's application is sent to a back-end server for processing. We apply stroke detection on this motion data to detect the start and end of a stroke. The magnetic data corresponding to the detected stroke is then fed into a multi-dimensional particle filter (Section 5.2), together with the writing behavior model. The writing behavior model helps in predicting the next state of the particle filter. The particle filter outputs the tracking traces of the stroke. An attacker then looks at the tracking result to guess what the user wrote.

5 SYSTEM DESIGN

This section describes the details for tracking when the users write with their natural handwriting style. To this end, we first describe our approach for building the Pencil's magnetic map. This is followed by the details of our tracking algorithm and stroke detection.

5.1 Pencil Magnetic Map Generation

The magnetic field map needed for tracking the Pencil movement is a function, f, mapping the location of the Pencil tip and its 3D orientation to the magnetic readings in x, y and z directions i.e.

$$(m_x, m_y, m_z) = f(x, y, orientation) \tag{4}$$

To build this magnetic map, we first want to introduce the coordinate systems for the iPad screen and the Pencil and clarify the notion of the Pencil's orientation.

Figure 7a shows the coordinate systems for the iPad screen, C_s. The top-left corner of the iPad screen is the origin for C_s. The Pencil tip location reported by the iOS touch API is also in the screen coordinate system. The

Z axis of C_s points downwards based on the right-hand rule. M_{loc} and T_{loc} are the locations of the magnetometer and Pencil tip in C_s, respectively. We assume that M_{loc} is known beforehand. Figure 7a also shows the Pencil's coordinate system, C_p. The Z-axis of the Pencil runs vertically through the Pencil's body. We define the orientation of the Pencil as its axes expressed as vectors in the C_s denoted by X_{ps}, Y_{ps} and Z_{ps}.

We can now define equation 4 as

$$(m_x, m_y, m_z) = f(T_{loc}, X_{ps}, Y_{ps}, Z_{ps}) \tag{5}$$

For the free-form writing case, we need to move the Pencil in all possible locations and orientations to get the magnetic impact for each location and orientation. This requires a huge amount of human effort. Hence, we focus on opportunities to reduce the war-driving space for building the magnetic field map.

So far, we have considered building the magnetic map from the iPad's perspective, i.e., the Pencil's location and orientation, and the magnetic readings are in C_s. Another way to look at this problem is to build the map from the Pencil's perspective. Recording the Pencil movement's magnetic impact on the screen is essentially the same as keeping the Pencil stationary and moving the magnetometer around it. In other words, for any position of the Pencil in C_s, we can find the corresponding magnetometer position in C_p. Given that we know the location of the magnetometer, M_{loc}, in C_s, if the Pencil tip is at location, T_{loc}, the 3D position of the magnetometer in C_p, M'_{loc} can be represented by

$$M'_{loc} = P_s * (M_{loc} - T_{loc}) \tag{6}$$

where P_s is the tuple (X_{ps}, Y_{ps}, Z_{ps}) representing Pencil's axes in screen space.

Similarly, we can also transform the magnetic readings in C_s, m to magnetic readings in C_p, m' by

$$m' = P_s * m \tag{7}$$

Building a map around the Pencil reduces the search space for war-driving since M'_{loc} can be the same for more than one Pencil position in C_s. This means that if we express equation 6 as

$$T_{loc} = M_{loc} - P_s^T * M'_{loc} \tag{8}$$

we can show that there are multiple pairs of T_{loc} and P_s which would correspond to the same M'_{loc} in C_p. Figure 7b shows an example of this case. Even though the T_{loc} and P_s are different for the two Pencils, M'_{loc} is the same for these two Pencils in C_p. Due to this redundancy, if we cover one of these Pencil positions while war-driving, we also determine the magnetic impact for a large number of other positions automatically. This greatly reduces the number of positions we need to cover through war-driving. Additionally, instead of collecting only one sample for each Pencil location and orientation, we can collect multiple samples for a position in C_p when we move the Pencil at different locations on the screen.

Considering the benefits mentioned above, we build the magnetic map around the Pencil, which can now be expressed as

$$m' = f(M'_{loc}) \tag{9}$$

For this purpose, we use our setup shown in Figure 7c to track the orientation of the Pencil while randomly drawing on the screen for a few hours. We record the Pencil movement with a camera placed at a distance above the iPad and looking down upon the screen. In this random drawing, we rotate and move the Pencil such that we can cover the entire screen and maximum possible orientations. Similar to the procedure for building the 2D magnetic map, we collect the magnetic data in the background at 50Hz and touch data, T_{loc}, from iOS API at 120Hz. The touch API only provides information about the azimuth and altitude angles [14] of the Pencil. However, the Pencil rotation around its Z-axis also affects the magnetic readings sensed by the magnetometer. Hence, the Pencil orientation, P_s, cannot be obtained without any ambiguity directly from the touch API. Therefore, we employ a computer vision approach to track the orientation.

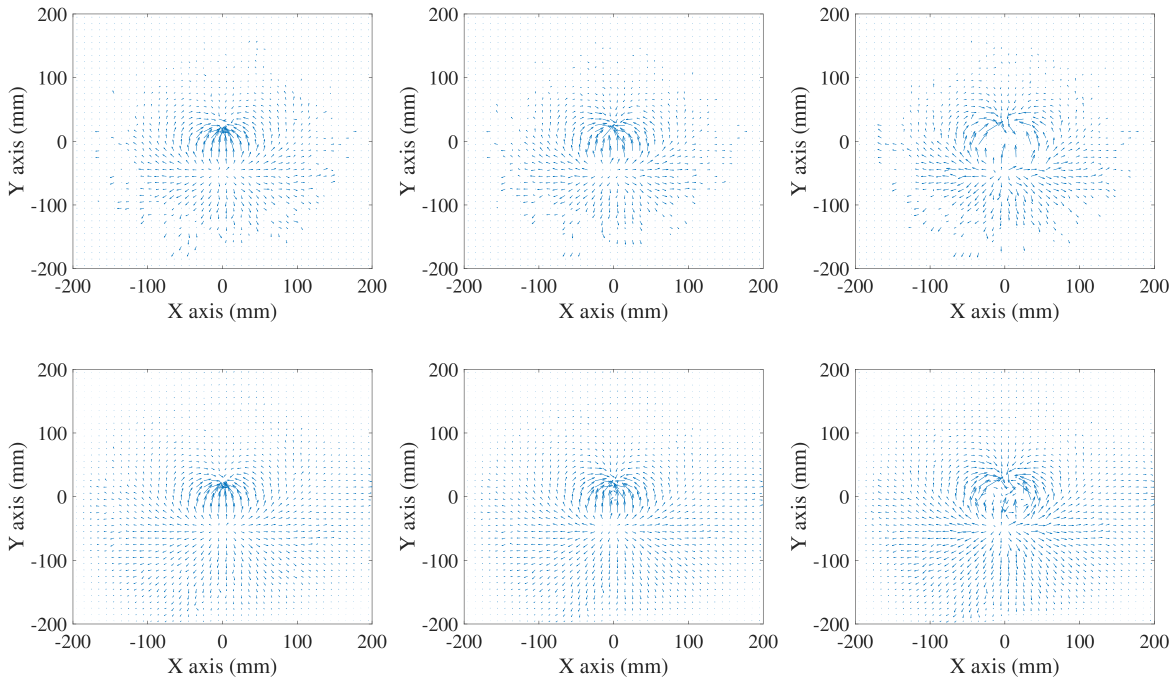


Fig. 8. Pencil magnetic map for $z = -5$mm (left) $z = 0$mm (center) and $z = 5$mm (right), before (top) and after (bottom) optimization.

The camera acts as a bridge between the screen and the Pencil. By finding the camera's orientation in C_s and then finding the Pencil orientation in the camera's coordinate system, we can finally find the orientation of the Pencil in C_s. To find the orientation of the camera in C_s, we use a standard camera calibration technique [15]. Once the camera is calibrated, we use PnP algorithm [16] to find the orientation of the checkerboard in the camera's coordinate system.

Due to the noise in camera calibration and pose estimation steps, the orientation obtained is noisy. To remove this noise, we apply Kalman filtering [8] and smoothing to the orientation obtained over time. Since the checkerboard's origin is attached to the Pencil end, the 3D axes of the checkerboard in C_s obtained after smoothing are indeed the Pencil axes in C_s as well. Therefore, by tracking the checkerboard axes, we can track the orientation of the Pencil as it moves on the screen.

We now have the touch data from the iOS API, the orientation data from the camera, and the magnetic data from the magnetometer. For each touch sample, we transform the M_{loc} and sample's corresponding m to C_p using equations 6 and 7 respectively. These transformed magnetic readings in C_p are finally quantized into $5mm$ by $5mm$ grids based on the 3D location of the M_{loc} corresponding to the reading. The average of all the magnetic data samples is computed for each grid. Figure 8 (top) shows the XY plane of this 3D map built around the Pencil for $z = -5mm$, $z = 0mm$ and $z = 5mm$ in C_p.

Here, we emphasize that we remove the magnetic impact of the ambient environment from the collected magnetic data before generating this map. As a result, the same magnetic map can be used to track the Pencil movement while the victim is using their iPad in different locations. Hence, this process is a **one-time effort** and does not involve any input from the victim. We also note that the setup shown in Figure 7c for Pencil's orientation tracking is only used for offline map generation and is not needed at attack time to infer victim's handwriting.

Map Optimization: As shown in Figure 8 (top), some of the grids in the 3D map have missing or noisy data. Instead of using a simple interpolation approach, we adopted a vector field reconstruction technique that considers the divergence and curl of the vector field [9]. It operates by minimizing the energy function J:

$$J(g; f) = \sum_{m} |Ag(m) - f[m]|^2 + \lambda_c ||curl(g)||_1 + \lambda_d ||div(g)||_1 \tag{10}$$

where g is the reconstructed map and f is the original magnetic map. A is an operator that only selects grids containing at least one datum. λ_c and λ_d are regularization parameters.

According to Maxwell's equations bounding the magnetic field [17], the regularization parameter λ_c should be small while λ_d should be large. Considering the noise in the magnetic readings, we perform a grid search to determine the optimal values of regularization parameters, guided by the optimal values given in [18] for different signal to noise ratios. The resulting 3D magnetic map from this approach is continuous with reduced noise. Figure 8 (bottom) shows the XY plane of this 3D map for $z = -5mm$, $z = 0mm$ and $z = 5mm$ after reconstruction.

5.2 Pencil Tracking

With the reconstructed 3D map, we are now prepared to describe our system for tracking the Pencil's location and orientation as a user writes on the iPad screen. Similar to the 2D case, we use particle filter to track the Pencil movement. However, now the state vector for the particle filter also has to take into account the Pencil's orientation.

Earlier, we defined the Pencil orientation as its axes, (X_{ps}, Y_{ps}, Z_{ps}) , in C_s. This definition is intuitive and works well for magnetic map generation but using this definition in particle filter is complicated. Another common way to represent the orientation of a rigid body is to use the Euler angles, which represent the 3D orientation of an object using a combination of three rotations about different axes [12]. Although intuitive, Euler angles suffer from singularities and give erroneous orientation representation when used to track an object's successive rotations. Fortunately, there is another formal representation for 3D rotation, which is the quaternions [12], which allows easier interpolation between different orientations. Quaternions encode any rotation in a 3D coordinate system as a four-element vector, where the squared sum of the four elements is equal to 1. Using quaternions, we can describe how the orientation changes sequentially more efficiently and smoothly [19]. Hence, for using the quaternion representation to describe the Pencil's orientation, our particle filter has to consider 6 parameters ($\langle x, y \rangle$ location of the Pencil tip and the Pencil orientation as 4-element quaternion vector) to track the Pencil movement, instead of 5.

To consider the orientation, we define the state vector for each particle at time t as:

$$s_t = < x_t, y_t, q_{1t}, q_{2t}, q_{3t}, q_{4t} > \tag{11}$$

Here, q_{1t}, q_{2t}, q_{3t} and q_{4t} represent the four scalar values for the quaternion representation of the Pencil's orientation at time t.

So far, we have considered that the Pencil can move in any orientation. However, in practice, the movement of the Pencil is limited by the range of motion of the human hand and wrist. Therefore, based on our observations of the human handwriting behavior, we limit the range for the Pencil's possible orientation. We assume that the altitude range of the Pencil is 30° to 90°. Similarly, for the azimuth angle, we specify the range to be within 60° to 170°. We do not limit the rotation of the Pencil around its Z axis. These ranges are determined by analyzing the minimum and maximum values of these angles observed in the data collected by the attacker and their accomplices for training the writing behavior model (described below). We specify the ranges for orientation

ALGORITHM 1: Pencil Tracking

```
1: Initialize particles $S_0^i$ where $i = 1, 2, ...N$ using orientation constraints
2: Compute the weights
3: Select top $K$ particles with the highest weights
4: for t ≥ 1 do
5:     Sample $S_t^i$ using $S_{max(t-3,1):t-1}^i$ based on writing behavior model
6:     Update particle history $S_{0:t}^i \leftarrow (\overline{S_{0:t-1}^i}, S_t^i)$
7:     Compute the weights
8:     Resample $S_t^i$ to obtain $K_a$ updated particles
9: end for
10: Track Pencil using $S_{0:t}^i$
11: function ComputeWeight(s)
12:     $P_s \leftarrow s$
13:     $M'_{loc} \leftarrow P_s * (M_{loc} - T_{loc})$
14:     Query Pencil magnetic map using $M'_{loc}$ to obtain $m'$
15:     $m \leftarrow P_s^T * m'$
16:     $w \leftarrow \exp(-\frac{(m-r)^2}{2\sigma^2})$
17:     return $w$
18: end function
```

in terms of the altitude and azimuth for ease of understanding, but these ranges are actually checked in the quaternion representation.

Algorithm 1 describes the algorithm for Pencil tracking.

Particles initialization: We begin by initializing the particles S_0^i where $i = 1, 2, ...N$. Since 6 parameters now determine the state, we need a huge number of particles to cover the entire state space. We employ a grid search approach on data collected from a small set of users (e.g., data collected for training writing behavior model which is not included in our evaluation set) to determine the optimal number of particles initially. To detail, we generate candidates from a grid of parameter values specified from 10000 to 10000000 with increasing the number of particles by 10000 in each iteration. We evaluate each of these particle values on the users' data and use the parameter that leads to highest accuracy at the elbow point [20] to prevent overfitting. Based on our search results, we set the number of particles, N, to 5000000. The x and y location for these particles are uniformly drawn from within the range of the iPad screen. Altitude (θ_1) , azimuth (θ_2) and rotation around Z axis (θ_3) are drawn uniformly from the range $30° - 90°$, $60° - 170°$ and $0° - 360°$ respectively. From these values for the three angles, we compute the axes representation for the Pencil in C_s. These axes are converted into quaternions and are included in the state vector. Once we have found S_0^i meeting our criteria, we compute their weights using the *ComputeWeights* function. For efficient computation, we specify a bound on the maximum number of particles at each timestamp as $K = 50000$. This bound is also found through the same grid search approach described above. Therefore, from S_0, we choose K particles which have the highest weights.

Movement model: We define the model for Pencil's movement for natural handwriting as:

$$s_{t+1} = s_t + v_t + b_t \tag{12}$$

where v_t is the change in particles' states, and b_t is the random perturbation added to the state. Previously, for the fixed orientation case in Section 3.2, we used $v_t = 0$ since the state space was small. For this case, we build a **writing behavior model**, which uses the changes in state from the last three timestamps to predict the state

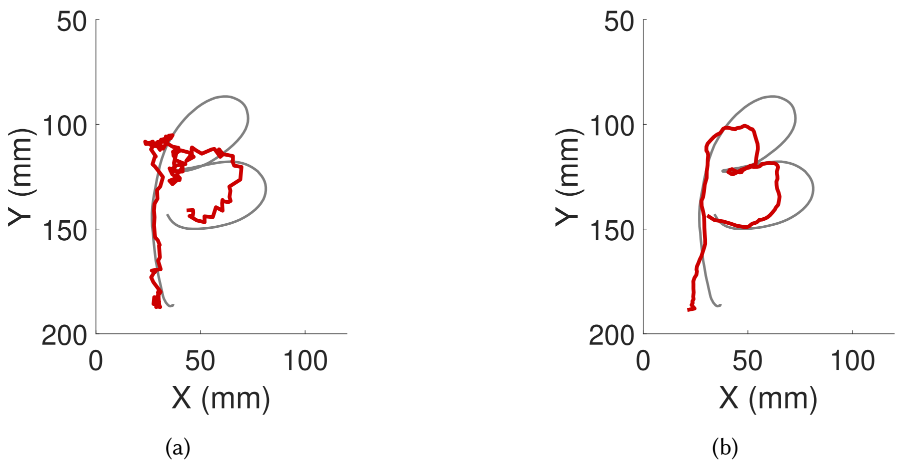


Fig. 9. Tracking results without (a) and with (b) writing behavior model.

change for current timestamps using linear regression. To train this model, we write different letters, numbers, and shapes at different locations on the screen. We use a camera to track the Pencil's orientation and iOS touch API to track location, as described in Section 5.1. This camera setup is only used while collecting training data for this model and is not used at attack time. Data collected from a small number of users (3 in case of our evaluation) is sufficient for training this model. We capture the relationship between location and orientation change in human handwriting from this location and orientation data. Figure 9b shows the result when this model is used in state transition. We can observe that the tracking result is more accurate and smoother than simply adding random perturbation to the last state (Figure 9a).

Computing particle weights: Here we transform the quaternions in the state vector to Pencil's axes, P_s, in C_s. We use these axes to find M'_{loc} using equation 6 in Section 5.1 to query our Pencil magnetic map. In this equation, T_{loc} is the x and y location in the state vector. The result of the query is m', which is the magnetic reading in C_p. This reading is converted to C_s using:

$$m = P_s^T * m' \tag{13}$$

Weights are assigned with the function:

$$w = \exp\left(\frac{(m-r)^2}{2\sigma^2}\right) \tag{14}$$

where r is the magnetometer reading corresponding to the given state.

Adaptive particle resampling: While a huge number of particles, in our case $K = 50000$, are needed to determine the initial Pencil position accurately, a smaller number of particles can suffice for tracking once the particles' states start to converge. For this purpose, we use the Kullback-Leibler distance (KLD) [21] resampling [13] approach to decide the number of particles, K_a, on the fly for every timestamp. KLD resampling operates by choosing the number of particles to resample such that the KLD between the particles' distribution before and after resampling does not exceed a predefined threshold. If the particles are widely spread, a large number of particles are used, whereas if the particles are more concentrated, a smaller number is resampled. We use a batched approach to increase the sample size for improving the efficiency of the resampling process. Figures 10a- 10d show how the resampling phase operates over time. In Figure 10a, the particles are scattered into two separate

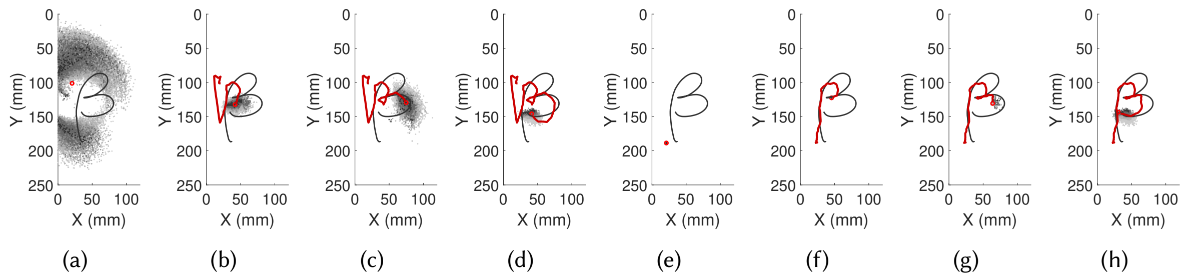


Fig. 10. (a-d) shows how adaptive particle resampling chooses the number of particles over time. (e-h) show the estimation of Pencil's location when history of particles is used.

clusters; hence a large number of particles is resampled. Overtime, once the particles start converging, the number of samples selected at each timestamp decreases as shown in Figure 10b- 10d.

Pencil location estimation: In addition to keeping track of the state, each particle also carries its history $S_{0:t-1}$ along with it for all timestamps $t \geq 1$. For the simple case of location tracking, we directly used the centroid of the particles to determine the Pencil's position at each timestamp. However, in this case, given the huge state space, the particles might exist in various clusters in different regions of the space, making it difficult to track using just the centroid. For example, in Figure 10a, there are two clusters of particles, and the centroid of these clusters (shown as red circle) does not match well to the ground truth Pencil location (i.e., bottom part of the character 'β'). In contrast, if we choose the maximum weighted particle in the last timestamp and use its history as the location estimate, we observe that all the particles converge at the beginning. This means that using the history of the particles returns us a path for the Pencil's movement with high accuracy. Figure 10e-10f shows the path retraced from the history of particles (red circle in Figure 10f).

5.3 Stroke Detection

To track a user's writing, we need to identify the precise period during which the user is writing from the collected motion sensors' data. To achieve this goal, we design a *two-step stroke detection* algorithm.

Step 1: Rough estimation: As shown in Figure 11(a), the magnetic field around the iPad will fluctuate when a user is writing, while it is stable when the Pencil moves away from the screen. Using this fluctuation, we first generate a rough estimate of the period when the user is writing. The basic idea is to use a sliding window with an appropriate size that contains a sequence of magnetometer data and move the window along the timeline. We mark the midpoint of this window as the starting timestamp of a stroke when the magnetic data variance is larger than a predefined threshold. Similarly, the ending timestamp is marked as the center of the window when the variance is below the threshold. We analyzed the data collected for training the writing behavior model for determining the optimal values for the window size and the threshold and set them to 100 and 0.12, respectively.

Step 2: Precise detection: A rough estimation is able to help us identify time periods which may contain a stroke. However, since the magnetic field starts fluctuating when the pencil is moving close to the iPad, it is difficult to identify the precise beginning and end of a stroke through magnetic data alone. Therefore, we utilize accelerometer and gyroscope data to detect the strokes more precisely. The intuition behind this step is that the iPad will vibrate slightly at both moments when the Pencil touches and is lifted from the screen. As shown in Figure 11(b) and (c), the beginning and end of the stroke are visible as prominent peaks in the accelerometer and gyroscope data. Hence the same threshold-based approach can be used on this data to identify the stroke with higher precision.

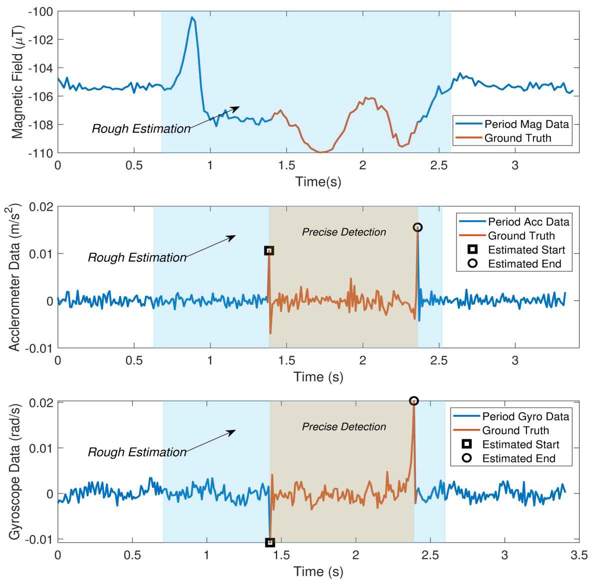


Fig. 11. Stroke detection to determine the beginning and end of a stroke.

6 EVALUATION

6.1 Data Collection and Implementation

We evaluated S^3 on an Apple 11" iPad Pro running iOS 12.0. We implemented two applications for conducting our experiments. The first application, A, acts as the malicious application installed on the victim's iPad, mimicking a legitimate application (such as a fitness tracker) that accesses the motion sensors' data and stays alive in the background (e.g., by use of location services). Application A logs the magnetometer readings at 50Hz, and accelerometer and gyroscope data at 100Hz in the background. The second application, B, mimics a chatting application with a text input region in the lower half of the iPad screen when it is used in landscape mode. This is where the user writes using the Apple Pencil. The input region is divided into 3 equally sized grids. This application acts as the application under attack from application A. The sensor data collected from application A is stored on the iPad during the data collection process and is transferred to a server for analysis. During our experiments, we observe that on average, with a fully charged battery, this application consumes less than 10% of the battery while logging the motion sensor's data, over a duration of 3 hours. We use our own custom application (application B) for writing to record the touch API data for ground truth. We also tested these applications on the latest iOS version (14.0) to validate our assumptions regarding required permissions.

Three authors (pretending to be attackers) collected magnetic data for the Pencil's magnetic map generation while randomly drawing on the iPad screen for a total duration of 3 hours. We used the setup shown in Figure 7c to record the orientation of the Pencil and the touch API's data for recording the Pencil's location. The three authors also wrote/drew the 26 lowercase alphabet letters (a-z), 10 numbers (0-9), and five different shapes (square, triangle, circle, heart, and star) twice in each grid in the input region of the application B. This data was used to train the writing behavior model described in Section 5 and was not a part of the evaluation set.

To evaluate S^3, we conducted experiments with 10 subjects, recruited by advertising on the university campus after IRB approval. These subjects included 6 females and 4 males. During the experiments, subjects were seated next to a table on which the iPad was placed. We asked each subject to use the Pencil to write the 26 lowercase

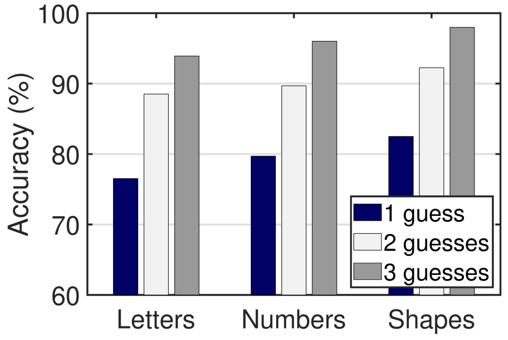


Fig. 12. Overall accuracy of the correctly guessed letters, numbers and shapes in 1, 2 and 3 guesses.

alphabet letters (*a*-*z*), 10 numbers (0-9) and five different shapes (square, triangle, circle, heart and star) twice in each grid in the input region. The subjects were guided to write in their natural handwriting style and we did not impose any restriction on the size of the strokes within the input region. In total, we collected the motion sensor data for 1560 letters, 600 numbers and 300 shapes.

In addition to letters, numbers, and shapes, we also tested our system's ability to track words. We asked five subjects to write words of lengths varying between three to six letters, where each word consisted of lowercase letters randomly selected from the English alphabet. We instructed each subject to write 30 words for each of the four word lengths. We also investigated our system's performance in detecting legitimate English words. For this purpose, we asked these subjects to write 30 words each, randomly selected from the 500 most commonly used English words [22]. These words represented a diverse set of words varying in length between 3 to 8 letters with an average length of 5 letters.

To demonstrate the practicality of our system, we also conducted experiments with different positions of the iPad and analyzed the impact of various environment settings.

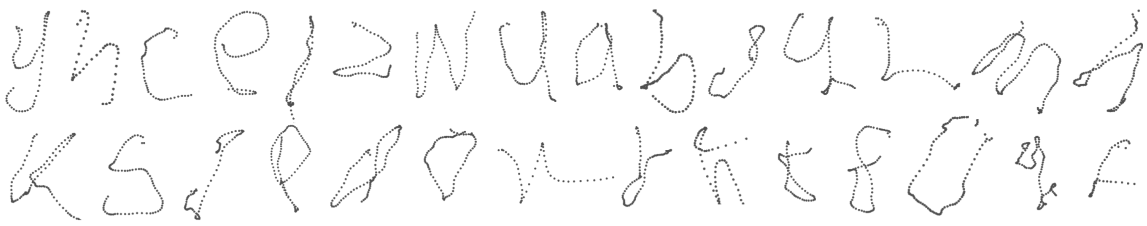
We randomly selected one of the volunteers to act as an attacker and showed S^3's Pencil tracking results to the attacker for each of our experiments. The attacker was given three guesses to identify each stroke. Hence our system's accuracy is determined as the number of strokes correctly identified by the attacker in the first three guesses. We use this top-*k* guess method considering the similarity in shapes of some letters in the English alphabet.

6.2 Performance Results

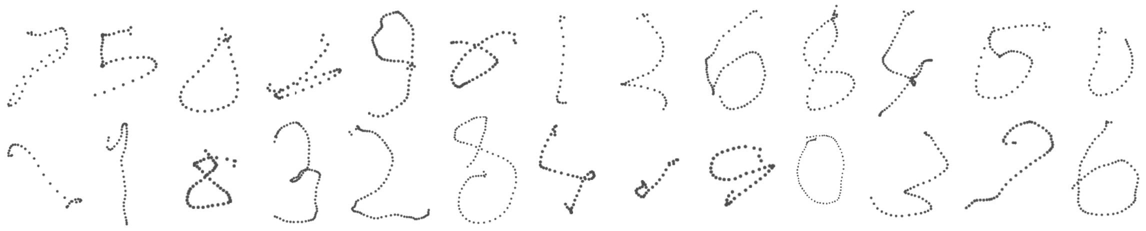
We answer the following questions to evaluate S^3's performance:

(1) How well can S^3 detect the letters, numbers and shapes?
(2) What is the detection accuracy for different letters, numbers, and shapes, and why?
(3) Is S^3 able to detect words?
(4) Can S^3 detect continuous strokes?
(5) How does the location of the Pencil tip affect the performance of S^3?
(6) Does the performance of S^3 change across users?
(7) How does the positioning of the iPad affect S^3's performance?
(8) How do the environmental factors impact the performance of S^3?
(9) How does S^3 perform compared to other machine learning techniques?

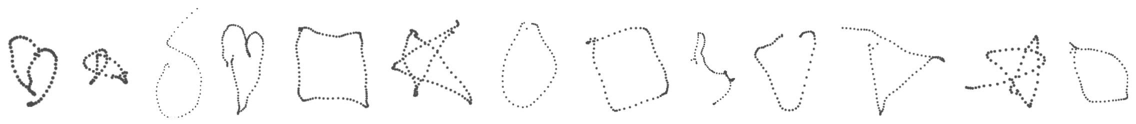
(1) How well can S^3 detect the letters, numbers, and shapes?



(a)



(b)



(c)

Fig. 13. Tracking result examples: (a) letters, (b) numbers, and (c) shapes.

We evaluated our system's performance by randomly selecting one of the volunteers acting as an attacker to guess what the users wrote from S^3's Pencil tracking results. The tracking results for different strokes were shuffled for this process to remove any possible bias. We recorded the top 3 guesses from the attacker for each stroke. Figure 12 shows the overall detection accuracy when the attacker's 1^{st}, 2^{nd} or 3^{rd} guess is correct for different strokes. The attacker correctly guessed 93.9%, 96%, and 97.9% of the letters, numbers, and shapes, respectively. This detection accuracy is based on whether any of the 3 guesses for a stroke matched with the actual letter, number or shape. Figure 13 shows a set of examples from the Pencil traces generated by S^3 solely using the motion sensor data. We encourage the reviewers to guess what letter, number, or shape is shown in each example. We give the correct answers at the end of this paper.

(2) What is the detection accuracy for different letters, numbers, and shapes and why?

We analyzed how different letters, numbers, and shapes contributed to the overall detection accuracy reported earlier. Figure 14 shows the detection accuracy for each letter, number, and shape based on the attacker's first guess. Here, we can clearly see that some letters, numbers, and shapes are detected better than the others in the first guess. For example, letters like 'b', 'e', and 'z' have a high detection accuracy. In contrast, letters like 'i', 'j', and 't' have lower accuracy. We found that these letters are usually written in two strokes. Since S^3 does not consider two strokes separately, the letter's shape is not obvious in the Pencil's trace. This can also be seen in Figure 13a, where it is difficult to recognize the letter 'j' (third trace in the bottom row). Apart from this observation, some letters like 'h' and 'n' have a low accuracy of 70% and 60% respectively because of the similarities in their shapes. The second letter in the top row of Figure 13a can be inferred as 'h' although it is actually 'n'. We observe similar patterns for 'r' and 'v' (67% accuracy for both) whose traces are similar to each other as shown in Figure 13a ('v'

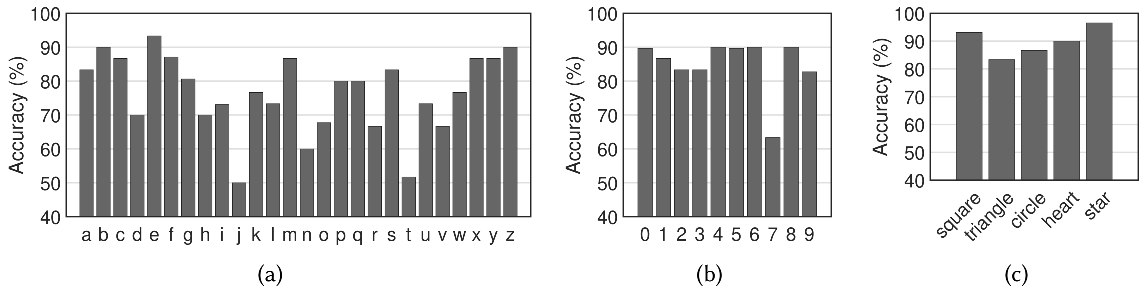


Fig. 14. Accuracy of the first guess for (a) each letter, (b) each number, and (c) each shape.

- third last letter in row 1, ‘r’ - 8^{th} letter in row 2). Similarly, the number ‘7’ can be confused with the number ‘1’. We note that the letters, numbers, and shapes whose traces are distinct from others are detected with high accuracy in the first guess. Examples of such cases are letters ‘m’, number ‘8’, and star shape.

Given that we did not impose any restriction on subjects’ handwriting styles, we observed that the same strokes written by the same subject often varied slightly in size or style (e.g., slanted). However, since S^3 does not rely on a specific handwriting style for accurate recognition, these variations do not affect its accuracy.

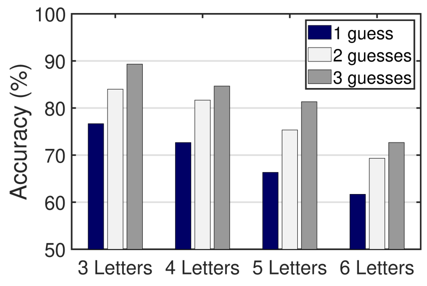


Fig. 15. S^3’s accuracy in detecting 3 to 6 letter words.

(3) Is S^3 able to detect words?

To answer this question, we shuffled the tracking results for all the words collected from the 5 subjects and presented them to the attacker as described previously. Figure 15 shows S^3’s detection accuracy for the random words of varying lengths. The attacker correctly guessed 89.3% and 84.67% of the 3 and 4 letter words. Given the completely random nature of these words, we observe that the number of correctly guessed words decreases as the words’ length increases. This is due to the similarity in the tracking results of different letters, as described above, which becomes more prominent in longer words. However, human handwriting in practice mainly consists of legitimate English words where little randomness is involved. For our experiments with the commonly used English words (three to eight-letter words), the attacker could guess the words with an accuracy of 93.33% in the first three guesses. Figure 16 shows a set of examples of the tracking results for this case. We found that the tracking results, in some cases, are noisy, making it difficult to guess all the letters in words. An example of this observation is seen in the rightmost word in Figure 16. Yet, the attacker can guess the words correctly in these cases since she is able to infer a few letters of the word from the tracking results, which allows her to figure out what the complete word is.

(4) Can S^3 detect continuous strokes?

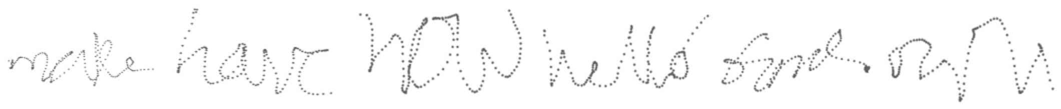


Fig. 16. Examples of tracking results for the words.

To evaluate continuous stokes, we conducted an end-to-end attack with S^3. To detail, we consider a real-world scenario where the victim uses the Apple Pencil to write a text message in a chatting application. The attacker's goal is to infer the text message (English sentences) based on the magnetometer's readings collected by the malicious application running in the background.

In these experiments, we invited three volunteers to write 10 different text messages on the iPad while sitting and holding it in hand. The subjects were asked to write valid English sentences, used in common text message conversations (without punctuation and emojis), with up to 8 words in each case. The number of words in the sentences written by the volunteers ranged from three to eight, with a median of five words. S^3's stroke detection component was used to separate the magnetic readings for different words within each sentence and fed into the particle filter. The attacker was able to guess 66.67% of all the words collected in the first three guesses through the tracking results (42.5% in the first guess and 59.1% in two guesses). We found that the estimated beginning and end of some strokes are not very precise when the time gap between different words while writing a sentence is small. Therefore, the tracking results for these strokes are noisy, making it difficult for the attacker to guess the correct word. However, the attacker was able to guess 80% of the sentences correctly when the tracking output of all the words in a given sentence were shown together as part of a single sentence. Thus, the context helps the attacker infer the complete sentence correctly even when the tracking results of individual words are noisy.

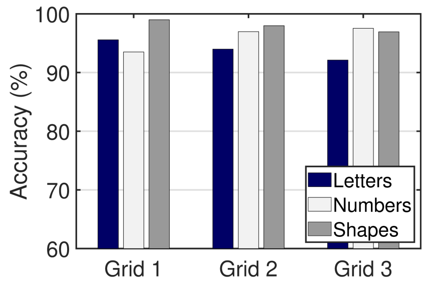


Fig. 17. Accuracy of detecting letters, shapes, and numbers across different locations on the iPad screen. Grid 1, 2, and 3 are the left, center, and right part of the input region.

(5) How does the location of the Pencil tip affect the performance of our system?

As mentioned earlier, we split the input region into 3 grids during the experiments. This enabled us to evaluate S^3's performance when a user writes on different iPad screen locations. Figure 17 shows the detection accuracy of letters, numbers, and shapes in the 3 grids. Since grids 2 and 3 are close to the magnetometer location, we observe that the range for magnetic fluctuations caused by the Pencil is large in these grids, whereas grid 1, which is farther from the magnetometer, is less sensitive to the Pencil's movement. However, we observe that our system performs consistently well across all grids despite detecting only small fluctuations in the grid 1. This observation clearly demonstrates that small changes in magnetic reading can be tracked by S^3.

(6) Does the performance change across users?

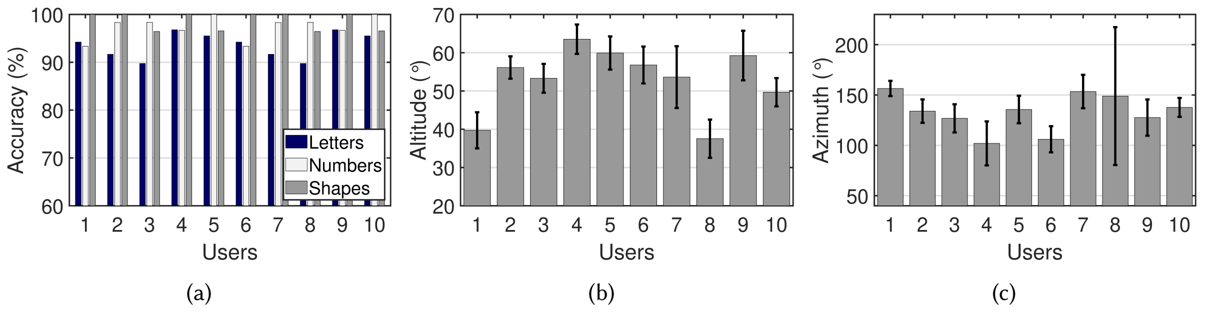


Fig. 18. (a) Detection accuracy of letters, shapes, and numbers for different users. (b) Distribution of the altitude and (c) azimuth angles for the strokes written on the screen.

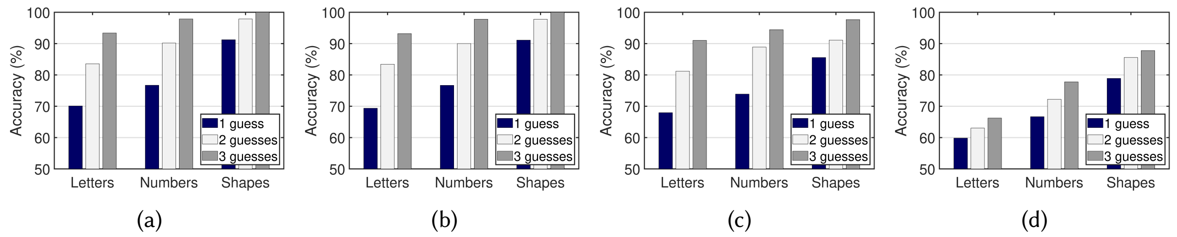


Fig. 19. Accuracy of the correctly guessed letters, numbers and shapes in 1, 2 and 3 guesses when the user is holding the iPad in hand (a) while sitting, (b) while standing, (c) while laying down, and (d) while walking.

We show in Figure 18a how well S^3 performs across different users. The detection accuracy is equal to or greater than 90% for letters, numbers, and shapes for all users. From the touch API data recorded for ground truth, we observe that the range for altitude and azimuth angles vary broadly across 10 users for each category. Figure 18b and Figure 18c show the average altitude and azimuth angles along with their standard deviation for each of the 10 users. Though we defined a fixed range for these two angles, the consistent good accuracy for all users shows that our tracking algorithm can accommodate a broad range of Pencil orientations and is not affected by the way a user holds the Pencil while writing.

(7) How does the positioning of the iPad affect S^3's performance?

We evaluated S^3's performance with a variety of iPad positions. Apart from the experiments with the iPad placed on the table, we conducted experiments with three volunteers in the following scenarios: 1) iPad is held in the hand while sitting, 2) iPad is held in the hand while standing, 3) iPad is held in the hand while laying down, and 4) iPad is held in the hand while walking. In each scenario, we asked the volunteers to write letters, numbers, and shapes on the iPad screen, as described earlier. Figure 19 shows the detection accuracy for letters, numbers and shapes in the top three guesses for each scenario. While a user is holding the iPad in hand, the writing causes small movements in the iPad body that introduce minor fluctuations in the motion sensors' data. However, a detection accuracy of higher than 90% for all strokes shows that S^3 is resilient to these small fluctuations and can still identify the Pencil's magnetic impact in scenarios 1, 2, and 3. For the scenario when the user is writing on the iPad while walking, we observe that S^3's detection accuracy reduces (66.2%, 77.8% and 87.8% for letters, numbers and shapes respectively) because the continuous change in ambient magnetic readings makes it difficult to separate the magnetic impact of the Pencil and the ambient environment.

Table 1. S^3's detection accuracy in different environmental settings.

Scenario	Letters (%)	Numbers (%)	Shapes (%)
Door opening and closing	93.37	95.55	96.67
People walking around in the room	93.80	96.58	97.86
Watch with metallic bracelet	91.88	95.55	96.67
Smart folio iPad case	91.67	94.44	96.67

(8) How does the environment impact the performance of S^3?

In our experiments, we observed that the magnetic field sensed by the iPad's magnetometer is not affected by regular objects such as books, clothes, etc., in the surroundings. The experiments described above were all conducted in a lab setting where electrical devices like computers, laptops, monitors, etc., were present in the surroundings. Based on these results, we concluded that the existence of these devices did not impact the performance of our system. Similarly, events such as opening or closing the door, people walking around, and the movement of furniture in the environment had no significant effect on S^3's performance.

We also evaluated how S^3's performance is affected by the presence of magnetic objects in the environment. We invited three volunteers to perform experiments in the following scenarios: 1) while wearing a watch with a magnetic bracelet on their non-dominant wrist, and 2) while writing on an iPad with a smart folio case. In both scenarios, the subjects were instructed to hold the iPad in hand. Table 1 shows our system's detection accuracy in the top 3 guesses for each scenario described above. In scenario 1, even though the magnetic bracelet is very close to the Pencil while the user is writing, its impact on the magnetic readings is much smaller than the Pencil's magnetic impact. As a result, despite the interference, S^3 is able to accurately track the Pencil position. In the case of the smart folio iPad cover, although the magnets in the cover affect the magnetic readings sensed by the iPad, their magnetic impact stays consistent over time, allowing S^3 to separate the impact of the Pencil as it moves over time.

(9) How does S^3 perform compared to other machine learning techniques?

To conduct a comparative evaluation with our system, we implemented three machine learning methods to infer the users' writing from the motion sensors' data. These methods included 1) k-nearest neighbors (k-NN) regression model, 2) Long Short Term Memory networks (LSTM) model for classification, and 3) an ensemble of LSTM models for classification trained for different locations of the iPad screen. We selected these algorithms as they capture the order dependence in our dataset and consider the nearest neighbors or previous states while guiding us about the pencil location over time (we detail the models in the Appendix.) The tracking results generated from the k-NN model allowed the attacker to guess only 11.54%, 18.67%, and 3% of the letters, numbers and shapes, respectively. The low accuracy is due to the fact that the magnetic readings can be similar for different Pencil locations and orientations and the k-NN model fails to capture the relationship between Pencil's previous states and its current location and orientation. Hence, the tracking results generated from this model are hardly legible. The LSTM model similarly achieved an accuracy of 7.69%, 10%, and 21.67% for letters, numbers, and shapes. This model fails to precisely infer useful information about the users' handwriting due to the variations in magnetic readings caused by changes in the Pencil location and orientation, even when a user is writing the same character (as illustrated in Figure 3), To guide the LSTM about the Pencil location, we also trained 3 separate LSTM models on data collected from each of the 3 grids in the input region. We observe that this approach slightly improves the LSTM model's accuracy and yields 11.54%, 16.67%, and 29.67% for letters, numbers, and shapes, respectively. However, this model also yields less accurate results than S^3's particle filter algorithm despite having a smaller search space for the Pencil's location. This is because this model fails to capture the combined impact of the changes in Pencil's location and orientation on the magnetic readings from the given data. In contrast to these models, S^3 can accurately track the Pencil's movement over time solely through the magnetic readings without requiring any large training dataset.

7 LIMITATIONS AND DISCUSSION

In this section, we discuss limitations and opportunities for the improvement of our system.

Limitations: S^3 currently can detect individual letters, numbers, and words. As we have mentioned in the evaluation, S^3 can infer sentences with a good accuracy when the tracking results of its words are analyzed together. Although this approach allows an adversary to infer commonly used English sentences through their semantic content, it might not be feasible for complex sentences when the semantic connections among words are lost or difficult to infer. A natural extension to our work would be to incorporate language models [23] for improving the detection accuracy of complex words and sentences. Since language models capture long-term dependencies, hierarchical relations and sentiment of the language, a sentence drawn from the modelled language, regardless of its complexity, can be inferred from an initial imprecise guess with a high accuracy.

We also demonstrated that S^3's performance is resilient to subtle changes in the environment. However, continuous changes in the ambient magnetic field, such as when a user travels in a vehicle, might interfere with Pencil's magnetic impact, making it difficult for S^3 to infer the users' writing.

In S^3's stroke detection process, more complex models through Recurrent neural networks (RNN) and Long-short term memory (LSTM) can be learned to capture the underlying sequence patterns in sensor data for detecting the beginning and ending timestamps of each stroke more accurately. Furthermore, our attack has a human in the loop for guessing what the victim is writing. Future work will expand our analysis to support models built on computer vision techniques to recognize letters, and natural language processing techniques (NLP) to infer words and sentences to infer user writings without human interaction.

Defense Techniques: We now discuss possible defenses for the side-channel attack presented in this paper. First, we observed in our preliminary experiments that if the sampling rate for accelerometer and gyroscope data is decreased, say to 50Hz, detecting the beginning and end of the strokes becomes very difficult. Similarly, if the magnetic data is sampled at a frequency of 5Hz, the tracking results become less tangible. Hence, a potential defense against our attack would require iOS to reduce the available sampling rate for accelerometer, gyroscope, and magnetic data when a user interacts with the iPad using Apple Pencil. Another potential defense would be to pause the motion sensors when a user interacts with the Pencil. However, both solutions heavily affect the operation of legitimate applications running in the background, which use motion sensors for activity, context, and gesture recognition. Another possible defense is to apply magnetic shielding [24] to the Apple Pencil. However, this will greatly impact the weight and hence the usability of the Pencil. Future work will analyze the trade-offs between these defense techniques and applications' access patterns to motion sensors.

8 RELATED WORK

Side-channel attacks through motion sensors: Several recent works have demonstrated potential privacy leaks from mobile sensors. Wang et al. explored how motion sensors' data collected from smartwatches are used to infer what a user is typing on a keyboard [1]. Another work demonstrated that the changes in the accelerometer readings are powerful enough to help extract passwords typed on smartphone touchscreens [4]. Michalevsky et al. showed the MEMS gyroscopes in modern smartphones could sense acoustic signals, which can be used to identify and parse speech [25]. A recent work unveiled a side-channel attack leveraging smartphone accelerometers to eavesdrop on the smartphone speaker and reconstruct the audio signals [26]. Das et al. [27] combined multiple motion sensors and used inaudible audio stimulation to fingerprint different users by measuring anomalies in the signals. Another line of work leveraged magnetometers to infer private information. It is shown that the magnetometer of a smartphone placed next to the hard drive of a computer allows an attacker to infer patterns about the system details, such as the type of operating system and applications used [28]. Block et al. leveraged magnetometers' ability to detect locations of users' devices within commercial GPS accuracy [29]. Researchers also demonstrated that electromagnetic field measured by smartphone magnetometers could be exploited for webpage [6], and device [30] fingerprinting. Compared with existing side channels, we introduce a new side-channel attack to identify users' handwriting by analyzing the magnetic impact of the embedded magnets in modern stylus pencils sensed by device magnetometers.

Handwriting tracking through motion sensors: There have also been efforts that explored the use of motion sensors for eavesdropping on users' handwriting. Motion sensor readings collected from a smartwatch are used to infer what the user is writing [31, 32]. However, these approaches require a compromised smartwatch to be worn on the victim's writing hand. In contrast, S^3 does not require any extra device, and tracks the Pencil movement with high accuracy without restricting the victim's handwriting style. Another work introduced *Finexus* to track fingertip movements in 3D space by instrumenting the fingertips with electromagnets and measuring the corresponding magnetic field changes using four magnetometers [33]. Although *Finexus* tracks the fingertip movements with millimeter level accuracy, it requires multiple magnetometers for precision. Our system tracks the pencil tip and achieves high accuracy in inferring handwriting with a single magnetometer. Similarly, *TMotion* presented a self-contained 3D input device that enables interactions in 3D space around smartphones by embedding a permanent magnet and an inertial measurement unit (IMU) in the stylus pen [34]. While this work is effective at tracking the stylus, it requires attaching an extra wand on top of the stylus pencil to accommodate a magnet and an extra IMU sensor, impacting the stylus pen's overall usability. In contrast, our system requires no such hardware for accurate pencil tracking. Lastly, a recent work introduced a sensing system for eavesdropping on handwriting by analyzing the magnetic field changes produced by stylus pens [35]. However, a commodity smartphone with a magnetometer must be placed within $20cm$ of the victim's device to sense the magnetic field. In contrast, S^3 uses the magnetometer on the victim's device, resulting in a higher detection accuracy, and does not require the attacker to be present in close proximity of the victim.

9 CONCLUSIONS

We introduced a side-channel attack through motion sensors by exploiting the vulnerability introduced due to the introduction of embedded magnets in stylus pencils. We presented S^3, a novel system that infers what a user is writing from motion sensors' data on an iPad Pro using the latest version of the Apple Pencil. To track the Pencil movement, we developed a novel multi-dimensional particle filtering algorithm using a 3D magnetic map of the Pencil to identify the magnetic impact of different locations and orientations of the Pencil. We evaluated S^3 with 10 subjects and demonstrated that an attacker could identify 93.9%, 96%, 97.9%, and 93.33% of the letters, numbers, shapes, and words correctly by only using the motion sensors' data. In future work, we will expand our analysis to support more devices that use stylus pencils with embedded magnets to find potential privacy leaks.

ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their insightful comments and valuable suggestions. This work has been partially supported by the Purdue Research Foundation (PRF) Research Grant.

Ground truth for examples in Figure 13 in order: Letters are y, n, c, e, l, z, w, u, a, b, g, q, v, m, h, k, s, i, p, d, o, x, r, h, t, f, j, q, and f. Numbers are 7, 5, 0, 6, 9, 6, 1, 2, 5, 8, 4, 5, 0, 3, 1, 8, 3, 2, 8, 4, 6, 9, 0, 3, 7, and 6. Shapes are heart, star, circle, heart, square, star, circle, square, heart, heart, triangle, star, and square.

Ground truth for examples in Figure 16 in order: make, have, now, hello, time, own.

REFERENCES

[1] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury, "Mole: Motion leaks through smartwatch sensors," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 155–166.

[2] Maryam Mehrnezhad, Ehsan Toreini, Siamak Fayyaz Shahandashti, and Feng Hao, "Touchsignatures: Identification of user touch actions and pins based on mobile sensor data via javascript," *CoRR*, vol. abs/1602.04115, 2016.

[3] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He, "Smartwatch-based keystroke inference attacks and context-aware protection mechanisms," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, New York, NY, USA, 2016, ASIA CCS '16, pp. 795–806, ACM.

[4] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang, "Accessory: password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 9.

[5] Rui Ning, Cong Wang, ChunSheng Xin, Jiang Li, and Hongyi Wu, "Deepmag: Sniffing mobile apps in magnetic field through deep convolutional neural networks," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 03 2018, pp. 1–10.

[6] Nikolay Matyunin, Yujue Wang, Tolga Arul, Jakub Szefer, and Stefan Katzenbeisser, "Magneticspy: Exploiting magnetometer in mobile devices for website and application fingerprinting," *arXiv preprint arXiv:1906.11117*, 2019.

[7] "Use apple pencil to enter text in any text field," https://support.apple.com/en-gb/guide/ipad/ipad355ab2a7/ipados.

[8] P. Zarchan, H. Musoff, American Institute of Aeronautics, and Astronautics, *Fundamentals of Kalman Filtering: A Practical Approach*, Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, Incorporated, 2000.

[9] Emrah Bostan, Pouya Dehgani Tafti, and Michael Unser, "A dual algorithm for L1-regularized reconstruction of vector fields," in *IEEE International Symposium on Biomedical Imaging (ISBI)*. 2012, IEEE.

[10] T.L. Chow, *Introduction to Electromagnetic Theory: A Modern Perspective*, Jones and Bartlett Publishers, 2006.

[11] Jeffrey Hightower and Gaetano Borriello, "Particle filters for location estimation in ubiquitous computing: A case study," in *UbiComp 2004: Ubiquitous Computing*, Nigel Davies, Elizabeth D. Mynatt, and Itiro Siio, Eds., Berlin, Heidelberg, 2004, pp. 88–106, Springer Berlin Heidelberg.

[12] James Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," 2006.

[13] Dieter Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, 2002, pp. 713–720.

[14] "Celestial coordinates," http://spiff.rit.edu/classes/phys373/lectures/radec/radec.html#altaz.

[15] Zhengyou Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.

[16] Xiao Xin Lu, "A review of solutions for perspective-n-point problem in camera pose estimation," in *Journal of Physics: Conference Series*. IOP Publishing, 2018, vol. 1087, p. 052009.

[17] James Clerk Maxwell, "Viii. a dynamical theory of the electromagnetic field," *Philosophical transactions of the Royal Society of London*, , no. 155, pp. 459–512, 1865.

[18] P. D. Tafti and M. Unser, "On regularized reconstruction of vector fields," *IEEE Transactions on Image Processing*, vol. 20, no. 11, pp. 3163–3178, 2011.

[19] Ramakrishnan Mukundan, "Quaternions: From classical mechanics to computer graphics, and beyond," 01 2002.

[20] Christopher M Bishop, *Pattern recognition and machine learning*, springer, 2006.

[21] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[22] "Word frequency data," https://www.wordfrequency.info/.

[23] Jeremy Howard and Sebastian Ruder, "Universal language model fine-tuning for text classification," 2018.

[24] Gary R. Scott, *Magnetic Shielding*, pp. 540–542, Springer Netherlands, Dordrecht, 2007.

[25] Yan Michalevsky, Dan Boneh, and Gabi Nakibly, "Gyrophone: Recognizing speech from gyroscope signals," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 1053–1067.

[26] Zhongjie Ba, Tianhang Zheng, Xinyu Zhang, Zhan Qin, Baochun Li, Xue Liu, and Kui Ren, "Learning-based practical smartphone eavesdropping with built-in accelerometer," in *Proceedings of the Network and Distributed Systems Security (NDSS) Symposium*, 2020.

[27] Anupam Das, Nikita Borisov, and Matthew Caesar, "Tracking mobile web users through motion sensors: Attacks and defenses.," in *NDSS*, 2016.

[28] Sebastian Biedermann, Stefan Katzenbeisser, and Jakub Szefer, "Hard drive side-channel attacks using smartphone magnetic field sensors," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 489–496.

[29] Kenneth Block and Guevara Noubir, "My magnetometer is telling you where i've been?: A mobile device permissionless location attack," in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2018, pp. 260–270.

[30] Beatrice Perez, Mirco Musolesi, and Gianluca Stringhini, "Fatal attraction: identifying mobile devices through electromagnetic emissions," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2019, pp. 163–173.

[31] Hao Jiang, "Motion eavesdropper: Smartwatch-based handwriting recognition using deep learning," in *2019 International Conference on Multimodal Interaction*, New York, NY, USA, 2019, ICMI '19, p. 145–153, Association for Computing Machinery.

[32] Q. Xia, F. Hong, Y. Feng, and Z. Guo, "Motionhacker: Motion sensor based eavesdropping on handwriting via smartwatch," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 468–473.

[33] Ke-Yu Chen, Shwetak N Patel, and Sean Keller, "Finexus: Tracking precise motions of multiple fingertips using magnetic sensing," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.* ACM, 2016, pp. 1504–1514.

[34] Sang Ho Yoon, Ke Huo, and Karthik Ramani, "Tmotion: Embedded 3d mobile input using magnetic sensing technique," in *Proceedings of the TEI'16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction.* ACM, 2016, pp. 21–29.

[35] Yihao Liu, Kai Huang, Xingzhe Song, Boyuan Yang, and Wei Gao, "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*, 2020, pp. 148–160.

[36] Francois Chollet et al., "Keras," 2015.

[37] Claude Sammut and Geoffrey I. Webb, Eds., *Leave-One-Out Cross-Validation*, pp. 600–601, Springer US, Boston, MA, 2010.

A DETAILS OF THE ML MODELS

K-Nearest Neighbors (k-NN) regression: We trained a k-NN model on the data collected for generating the magnetic map for the Pencil and evaluated it on the data collected from the subjects in our user study. The model was designed to predict the Pencil's location and orientation at time t when given the magnetic readings for the previous three timestamps. We used a grid search to find the best value for k within the range of 1 to 5 on our training data and chose the value which resulted in the highest accuracy. The attacker was shown the tracking results generated from predicted Pencil location trace for each stroke.

Long Short-term Memory Network (LSTM): We implemented a LSTM model for predicting which letter, number or shape corresponds to magnetic readings collected for a stroke. We used TensorFlow's Keras library [36] for implementing the LSTM network. The model required a three-dimensional input of the shape [samples, time steps, features] where samples is the number of strokes, time steps is the number of magnetic samples for each stroke (fixed to 200 samples) and the number of features is 3 representing the x, y and z axis of the magnetometer readings. The output of the LSTM model was a 26 element vector (10 and 5 in the case of numbers and shapes respectively) representing the probability of a given stroke being any of the 26 letters (10 numbers or 5 shapes). We defined the model as sequential model with two LSTM hidden layers followed by a dropout layer to reduce over-fitting on training data. The features extracted from the LSTM layers were fed into a dense fully connected layer followed by a final output layer used to make predictions. We trained this model on the data collected from the 10 subjects in our user study using a leave-one-out cross validation approach [37].

In an attempt to improve the LSTM network's performance, we limited the search space for the Pencil's location by implementing separate models for each of the three grids in the input region of the screen. The same network architecture was used for the three models. For final inference, we used the output of the model with the highest confidence score.