Weaponizing IoT Sensors: When Table Choice Poses a Security Vulnerability

Gustavo Casqueiro*†
Military Institute of Engineering
Rio de Janeiro, RJ, Brasil
gustavo.casqueiro@ime.eb.br

Sayed Erfan Arefin†
Texas Tech University
Lubbock, Texas, USA
saarefin@ttu.edu

Tasnia Ashrafi Heya
Texas Tech University
Lubbock, Texas, USA
tasnia.heya@ttu.edu

Abdul Serwadda
Texas Tech University
Lubbock, Texas, USA
abdul.serwadda@ttu.edu

Hassan Wasswa
Texas Tech University
Lubbock, Texas, USA
hwasswa@ttu.edu

***Abstract*—The security threat posed by keyloggers on laptop and desktop computers is traditionally understood from the perspective of malware that directly reads keystrokes on the victim's machine. While recent research on smart phone platforms has shown that motion/vibration sensors inbuilt in these phones also pose a keylogging threat, this line of attack has never been investigated in desktop and laptop settings given that no such sensors exist in these settings. In this paper, we show that the vibration dynamics of commonly used computer table materials transmit keyboard vibrations during typing with such fine granularity that keyboard typing locations (and hence keystrokes) could be learned from the vibrations. In practice such an attack would be executed by methodically rigging the underside of a computer table or keyboard itself with a series of motion sensors, and then mining the data generated by these sensors during typing. Taking the case of typical computer table materials such as glass, plastic, metal and wood, we study this line of attack and highlight scenarios where it poses a potent threat. Thanks to fast growing IoT platforms making available easy-to-use, fully featured, cheap sensors, we argue that this line of attack is accessible to even casual "computer hackers" having no knowledge of low-level hardware programming. The paper brings to light a previously unexplored privacy threat that security practitioners and end-users need to pay attention to as IoT goes mainstream.**

***Index Terms*—internet of things, sensors, side-channel attack, machine learning, keylogging, keyboard inference attack**

I. Introduction

Imagine the following scenario. Eve wants to spy on the messages that her boss Bob types on his desktop keyboard. Eve however does not have login credentials to Bob's system, and hence cannot install a key logger on it. Eve opts to buy several motion/vibration IoT sensors and *methodically "plants" them under Bob's desktop keyboard or under the table on which the keyboard sits*. She will then analyze the microscopic vibrations captured by these sensors while Bob types and hopefully leverage them to infer the text typed (e.g., emails, PINs, passwords,etc.).

While keystroke side-channels have seen a significant amount of recent research, the line of attack articulated above has surprisingly never been studied. A possible reason behind this is that threats from motion-sensors are typically imagined in the context of smart phones, which come inbuilt with these sensors. With the proliferation of IoT technologies and devices in recent years however, fully-featured, plug-and-play, sensors of all kinds are available on the open market, and could be deployed in all kinds of settings beyond the now well-understood smart phone scenario.

For example, an entity seeking to execute the attack described above can easily buy a series of vibration sensors (e.g., see [1]) that could easily be embedded in any device earmarked for the attack. Such sensors are custom built for the average "Do-It-Yourself" (DIY) customer who has no knowledge at all of hardware programming. At a price point of less than $50 (see [1]), each of these sensors comes equipped with an accelerometer, gyroscope, and, magnetometer, and includes support for Bluetooth connectivity with up to 50 metres range. All that the attacker needs to do is follow the manual and plug the various cables in the right places before launching the associated mobile or desktop apps that come with the sensors for data collection. Before IoT became mainstream and launched these kinds of sensors onto the open civilian market, an adversary seeking to execute this attack would have needed significant low-level hardware programming knowledge. This should in turn have limited the likelihood that one could ever be subjected to this kind of attack. With the IoT revolution making these kinds of attacks much more likely, it is instructive that these IoT-driven attacks be studied so that the broader research community can understand their dynamics and possibly explore defense mechanisms. *The previous statement underlines the primary goal behind of this paper.*

Taking the case of PINs typed on the numeric pad of a standard Logitech keyboard, we investigate the question of whether a series of vibration sensors hidden under the keyboard, or under the table holding the keyboard could be

†These co-authors contributed equally.

*This research was conducted while he was visiting Texas Tech University in the Fall of 2021.

used to infer text typed on the keyboard. We use a commodity vibration sensor bought off of Amazon for the attack (see Figure 2), and take advantage of the small size of this sensor (i.e., 1.5cm) to rig as many as 4 sensors under the table. We study this attack scenario for various types of table surfaces and highlight cases for which the attack poses a serious threat.

The contributions of our paper are summarized below.

1) **Design and evaluation of an attack in which underside of keyboard is rigged with motion sensors:**. We present an attack in which an adversary attaches a series of motion sensors under a computer keyboard with the aim to infer text typed on the keyboard. We execute this attack on PINs typed on the keyboard and show it to be highly effective, attaining a classification accuracy of up to 90% at the inference of individual keys. We perform a sensitivity analysis on the attack, highlighting how variables such as the numbers of sensors and distance of keys from the sensors impact attack performance.
2) **Investigation of attack behavior for sensors hidden under tables made out of four commonly used materials:** In cases where the attacker might need extra stealth or where the specific design of a keyboard might not enable the insertion of sensors under it, the attacker could opt to put the sensors under the table, right below the keyboard. We also design and evaluate this line of attack, studying four commonly used table-top surfaces (namely, wood, glass, metal and plastic). On one extreme we find the plastic table to be highly vulnerable to the attack, and on the other, the wooden table to completely thwart the attack. The other two surfaces depict a behavior in between these two extremes.
3) **Investigation of impact of table rotation and translation on attack behavior:** For the attack variant in which sensors are hidden under the table, it is possible that the victim might move the keyboard during their routine activities, and inadvertently thwart the attack after mis-aligning the keyboard with the sensors. We run experiments where the keyboard undergoes small translations and rotations relative to its initial position, and show that these can be very easily detected from sensor data features. This result implies that an attacker who detects such movements can abort the attack until he/she gets the next opportunity to access the victim's space and re-align the sensors.

Road-map: The rest of the paper is organized as follows. We present our threat model and attack design requirements in Section II. We then present the attack design and evaluation in Sections III and IV. Afterwards, we discuss related research in Section V and our conclusions in Section VI.

II. Threat Model

Our attack would typically be executed by an insider attacker who has frequent access to the room containing the victim's computer. People who might have such access to the victim's space include work colleagues, office cleaners, roommates, spouses, employers, etc. The attacker seeks to spy on the victim's keyboard inputs, but has no login (or root) access to the victim's computer/laptop and is thus unable to install malware to undertake this operation. The attacker then opts to rig the keyboard, table on which the keyboard is placed, or laptop stand with motion and vibration sensors. The attacker's frequent access to the victim's space ensures that there are ample opportunities during the victim's absence from the room for installation, adjustment (if needed) and retrieval of the sensors.

To generate data for training of the machine learning models, the attacker has two options: (1) replicate the victim's setup by acquiring equipment (e.g., keyboards, tables, etc.) similar to those of the victim so as to closely match the vibration behavior of the victim's system during typing, or, (2) use the victim's very system for typing the relevant training data. The second option is possible because the generation of training data does not require one to log onto the system. All that is required of the attacker and (or) his/her accomplices is to type content on the victim's keyboard while sensors under the keyboard/table capture the corresponding data. Again, the attacker's unfettered access to the victim's space allows many opportunities for this kind of data collection. For example, a spouse, roommate or office cleaner would have many hours per day to execute the training data collection using the victim's very system.

During the attack itself, the attacker could leverage various contextual cues to increase the odds of attack success. For example, by virtue of being an insider attacker, the attacker might have a good idea of when the victim might type the information being targeted for attack (e.g., a spouse might have an idea of the times when the husband might be chatting with his mistress(es)). This way, the attack could be focused on carefully selected time-windows of sensor data. Even where the attacker might not have such information, well-known computer access patterns might help focus the attack. For example, if the attacker's target is to steal login credentials, the attacker could exploit the fact that login details are the first text that the victim enters on the keyboard in the morning, or later in the day after breaks from work (e.g., lunch break). The first few seconds of sensor data collected after long pauses in sensor vibrations would thus be the focus of analysis in such a case. Figure 1 shows a high-level view of the attack process. While our attack implementation used Bluetooth to send sensor data to the attacker's workstation, one could, depending on sensor features, opt to store sensor data on a memory card within the sensor and retrieve it later, or even use Wi-Fi to enable long-range data transmission.

III. Attack Design and Implementation

A. Hardware Used

The sensors used in our attack are Witmotion's WT901BLECL model [1]. Each sensor has a 3-axis accelerometer and gyroscope and also measures angles and magnetic fields. Using software that ships with the sensors, up to 4 of these sensors can be connected to an Android device or Windows PC for real-time data transfer via a Bluetooth

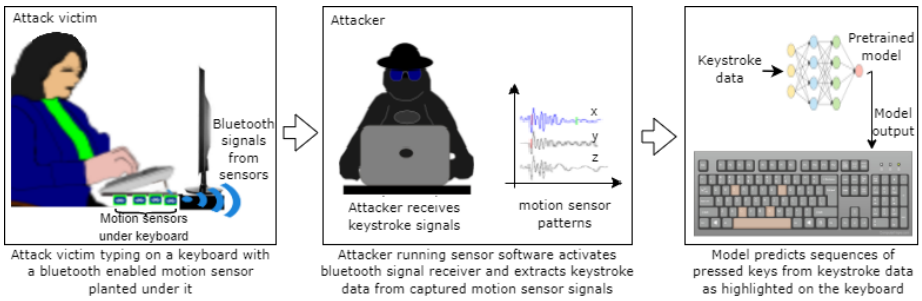


Fig. 1: Overview of attack process.

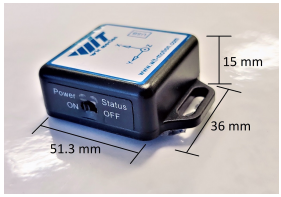
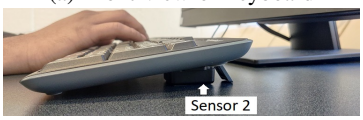


Fig. 2: One of the sensors used in our experiments

(a) Front view of keyboard



(b) Side view of keyboard

Fig. 3: Sensors hidden under the keyboard.

connection having 50 meters range (if no walls). The sensor is shown in Figure 2. The 15mm height/thickness enables it to stealthily attach under the raised end of a desktop keyboard without any noticeable impact to how the keyboard sits on a table surface (see Figure 3). One could remove the casing and further reduce the sensor thickness and other dimensions in order to push the sensors deeper under the keyboard. This would hide the sensors better and also enable the usage of more sensors under a given region if needed.

The keyboard used in our experiments is the Logitech MK270 wireless keyboard [2] (also shown in Figure 3). The sensors were attached under this keyboard using double-sided tape. Note that the pictures in Figure 3 were taken with such a camera angle to emphasize the sensors for illustration purposes in this paper. Otherwise in many practical scenarios the sensors would likely go unnoticed given that they occupy a small portion of the keyboard width which is partly obfuscated by one of the keyboard legs and the computer monitor. For victims who work in settings where the backside of the computer monitor faces a wall (e.g., our own research lab and similar shared office spaces), or where the work-table has books, gadgets and other clutter, the odds of the sensors being seen would even get lower. Finally, we note that in the experiment with sensors under tables not made out of glass (details of all our experiments are in Section III-B), the idea of the sensors being seen would even be out of question given realistic assumptions.

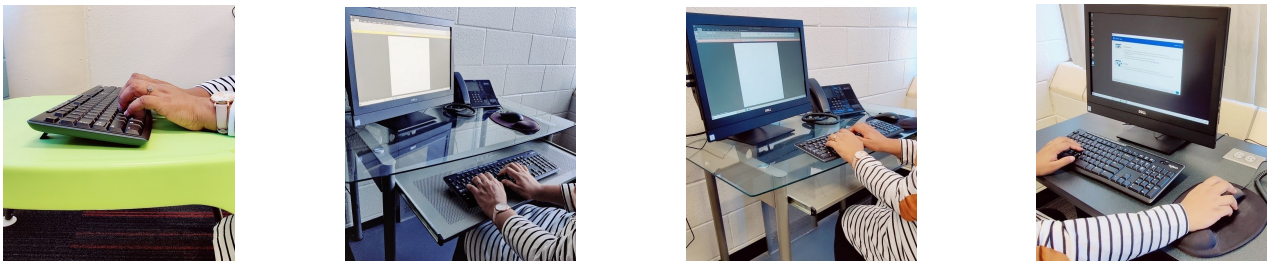
B. Data Collection Experiments

Human subjects considerations: Data collection under this research was approved by our university's IRB. All experimenters involved in data collection took the required IRB training. All user data was not linked at all to the participant identities. Our experiments imposed no privacy or security risks to the participants since all they did was type random passwords generated specifically for the study. All experiments were done on our lab computers; there was no software installation of any sort on participants' devices.

Experiment details: Our experiments focused on numeric inputs typed on the keyboard numeric pad. We specifically collected data for two variants of the attack: (1) Attack with sensors under the keyboard — In this attack we used two sensors attached under the numeric pad of the keyboard as shown in Figure 3, and, (2) Attack with sensors under the table — In this attack, we used four sensors attached under a table on which a keyboard sits. Four different table materials were used, namely, wood, glass, metal and plastic (see Figure 4). The sensors align with the numeric pad region of the keyboard as illustrated in Figure 5.

In all our experiments, the sensors are placed under the numeric pad because we simulate an attacker who targets this pad to specifically steal numeric inputs (PINs, SSNs, monetary figures, e.g., typed by bankers, etc.). This design is a case study of the broader family of attacks which might leverage the flexible sensor placement supported by this attack to target different keyboard regions depending on what information one wants to steal or application one wants to spy on.

Our data collection involved 23 participants across all experiments. For the main number classification experiment, participants provided training data by typing the digits 0-9, 30 times each on the number pad. For testing, participants typed full PINs that comprised a mix of 4, 6 and 8 digit PINs with 5 PINs of each length. In total, each participant typed 15 PINs. The PINs were randomly generated from this



(a) Plastic table (b) Metallic table (c) Glass table (d) Wooden table

Fig. 4: A participant keyboarding on tables under which sensors have been attached. The four tables represent the four different table materials used in our experiments. For locations of the sensors under the tables, see illustration in Figure 5.

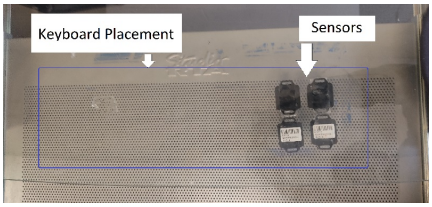


Fig. 5: Sensor placement under the glass table. The keyboard has been removed to reveal the sensors under the glass. The blue line shows the area where the keyboard was placed so as to align the sensors with the numeric pad. The sensors and keyboard were positioned similarly for the other three table surfaces.

website [3]. For the experiment to detect keyboard translation or rotation from the initial position when sensors were under the table, participants typed PINs just like in the main number classification experiment.

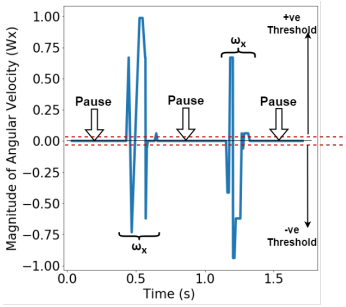


Fig. 6: X component of gyroscope sensor data generated after two consecutive key strokes on the keyboard.

C. Data Processing and Feature Extraction

Our sensors have an embedded Kalman filter, hence the data collected from them was already quite clean requiring no noise removal by us. Data generated by the sensors was taken directly through a character segmentation step that extracted sensor data segments that are delimited by pauses such as those illustrated in Figure 6. The figure shows data from the X axis of the gyroscope sensor when the key-pair 1 and 2 was typed. In general, before a key is typed, sensor readings are almost zero. When a key is typed, there is a brief spike in the data which again dies down to about zero after key release. From the three sensor axes (Y and Z not shown in the figure) we computed the magnitude of the angular velocity which we then used as a basis to cut out pauses. Having segmented the characters, we extracted features from each of the 6 sensor data streams (3 accelerometer axes and 3 gyroscope axes) for every 2-second window of data. For this we used *tsfresh* [4] library which generates an assortment of time and frequency domain features from time-series data.

The features listed in Table I were calculated for each data window and each axis. A detailed description of each of these features is provided in the library documentation, hence we do not repeat this here.

When *tsfresh* feature selection function [5] was used, the classification results in our proof-of-concept experiments were poorer. The same happened when PCA was used. We ultimately just did a basic data cleaning on the features — i.e., we dropped features which had a constant value as well as missing (NaN) or infinity (INF) values.

D. Machine Learning Models

For each our two overarching experiments (recall keystroke inference, and keyboard movement detection in Section III-B), we built machine learning models for different data input scenarios. Table II summarizes these scenarios. The following two sentences provide examples on how to interpret Table II. In the keystroke inference attack, when the sensors were under the keyboard, we had models which were built using data from Sensor S1 alone, Sensor S2 alone, and the combination of S1 and S2.

For each sensor data stream we built two kinds of machine learning models: one based on an individual classifier, and another based on a voting ensemble. All classifiers are run with python's *scikit learn* library [12]. The individual classifiers were always *XGBoost* or *LightGBM* (depending on which performed better on preliminary data). These two were chosen because they generally outperformed all other individual classifiers that we explored during the research. The voting ensemble is scikit learn's *VotingClassifier* [13] which provides an interface for combining and tuning fusion configurations of individual classifiers. Table III gives details

Features				
Skewness	Sum of reoccurring values	Approximate Entropy	Has Duplicate	Last Location of Maximum
Sum values	Sum of reoccurring data points	Max Langevin Fixed Point	Abs Energy	Last Location of Minimum
Root mean square	Ratio value number to time series length	Variation Coefficient	Fourier Entropy	First location of minimum
Length	Variance Larger than Standard Deviation	Ratio Beyond R Sigma	Standard Deviation	First location of maximum
Kurtosis	Time Reversal Asymmetry Statistic	Mean Abs Change	Number Peaks	Absolute maximum
Maximum	Mean Second Derivative Central	Partial Auto correlation	Auto Correlation	Count Above Mean
Median	Longest Strike Above Mean	Number Crossing M	Has Duplicate Max	Variance
Mean	Longest Strike Below Mean	Mean N Absolute Max	Has Duplicate Min	Mean Change
Quantile	Large Standard Deviation	Agg Autocorrelation	Number Cwt Peaks	C3
Cid Ce	Augmented Dickey Fuller	ABS Sum of Changes	Count Below Mean	Benford Correlation
Minimum	Query Similarity Count	Lempel Ziv Complexity	Range Count	Binned Entropy

TABLE I: Features calculated for all the sensors using the *tsfresh* library [4].

Experiment		Placement	Sensors used
Keystroke Inference Attack		Under the keyboard	S1
			S2
			S1+S2
		Plastic Surface	S1
			S2
			S3
			S4
			S1+S2+S3+S4
		Glass Surface	S1
			S2
			S3
			S4
			S1+S2+S3+S4
		Metallic Surface	S1
			S2
			S3
			S4
			S1+S2+S3+S4
Detecting Keyboard Movements	Rotation	Plastic Surface	S1+S2+S3+S4
	Translation	Plastic Surface	S1+S2+S3+S4

TABLE II: The different data input scenarios used to build our machine learning models.

of the constituent classifiers and associated parameters for the voting ensemble classifier which was built for the case when the sensors were placed under the table, and the input data stream was S1+S2 (recall Table II). This particular ensemble was selected after searching a wide range of configurations from which we selected the best performing one. We leave out the detailed parameter-sets for the other scenarios, but provide a high-level summary of the number of instances of each classifier used in the ensemble (see Table IV). The table particularly shows the cases with input data from 4 sensors (i.e., cases of table surfaces) and 2 sensors (case with sensors under keyboard).

For the individual classifiers, Table V shows the classifier details for the keystroke inference experiment. The details shown apply to all cases (i.e., models from input streams S1, S2, and S1+S2).

IV. Attack Performance Evaluation

In the following sub-sections we present results from all our attack scenarios.

A. Keystroke Inference Attack

Figure 7 shows the keystroke inference accuracy for the case with the sensor under the keyboard and the cases where the sensors were under the 3 types of tables. In each case we show results when data from one sensor is used to build machine learning models that run the attack and where data from all sensors is used. In all cases we see that for either classifier, performance improves when multiple sensors are used. This is unsurprising given that more sensors should cover a wider area and in turn better represent the vibrations with finer granularity.

The figure also reveals that the case with the sensors under the keyboard performs best (Figure 7a), as it reaches an accuracy of 90% in the best case. This accuracy is 9-times the random guessing accuracy of 10% for the 10-class problem. This good performance is likely because the direct contact between the sensors and the keyboard enables unadulterated transfer of vibrations to the sensors, which in turn allows the classifiers to learn the underlying patterns more rigorously.

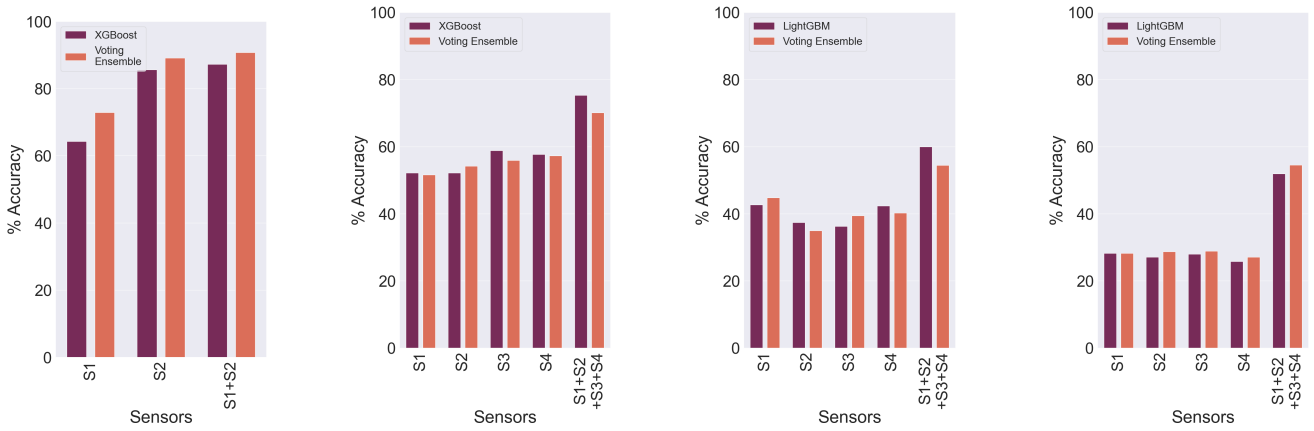
Of the three tables, the attack performs best with the plastic table and worst with the metallic table. That said, even the worst performing table scores a little better than 50% accuracy when all sensors are used, a performance way higher than random guessing.

The confusion matrices in Figures 8 and give us some insights into where the classification errors came from. The matrix in Figure 8 is for the scenario with the sensor under the keyboard and the Voting Ensemble classifier. Observe that in general, the numbers in the matrices tend towards zero as one moves from the main (dark blue) diagonal towards the bottom left corner. The same happens as one moves from the main (dark blue) diagonal towards the top right corner. This pattern indicates that misclassifications are for the most part between neighbouring keys. For example, in figure 8, the key, 1 is misclassified as 0 with probability 6.6%, and as 2 with probability 3.8%. The key 1 does not see any other significant misclassifications beyond these two. In the context of the attack, this pattern means that any misclassifications made by our attack could be fixed by searching a narrow set of keys in the neighbourhood of the wrong prediction. This same general pattern was seen with the other attack scenarios, further pointing to the lethality of the attack.

Our final analysis of the keystroke inference attack is

Classifier	Parameters	Data Transformation
XGBoost [6]	colsample_bytree: 0.8, eta: 0.3, max_depth: 10, max_leaves: 0, n_estimators: 100	Standard Scaler
XGBoost	colsample_bytree: 0.5, eta: 0.2, max_depth: 6, max_leaves: 0, n_estimators: 400	Standard Scaler
XGBoost	colsample_bytree: 0.7, eta: 0.3, max_bin: 63, max_leaves: 7, n_estimators: 600	Standard Scaler
Light GBM [7]	colsample_bytree: 0.69, max_bin: 190, min_child_weight: 0, n_estimators: 200	Max Abs Scaler
Light GBM	colsample_bytree: 0.199, max_bin: 230, min_child_weight: 1, n_estimators: 800	Robust Scaler
Random Forest [8]	Scikit-learn defaults	Standard Scaler
Extra Trees [9]	criterion: gini, max_features: 0.9, min_samples_leaf: 0.01, n_estimators: 25	Min Max Scaler
KNeighbors [10]	n_neighbors:10	Sparse Normalizer
SVM [11]	C: 2222.9, kernel: rbf, class_weight: balanced	Standard Scaler
Light GBM	min_data_in_leaf: 20	Max Abs Scaler

TABLE III: Classifiers parameters and data transformation algorithms used for the experiment where sensors were under the keyboard for the keystroke inference attack.



(a) Sensors under keyboard (b) Sensors under plastic table (c) Sensors under glass table (d) Sensors under metallic table

Fig. 7: Keystroke inference performance on the 10-class problem involving the numbers 0-9

Classifier used with Data transformation		Attack Surfaces			
Classifier	Data Transformation	UK	PS	GS	MS
Light GBM	Standard Scaler	0	0	3	3
Light GBM	Min Max Scaler	0	0	1	1
Light GBM	Max Abs Scaler	2	1	1	1
Light GBM	Robust Scaler	1	1	0	0
XGBoost	Sparse Normalizer	0	0	2	2
XGBoost	Standard Scaler	3	5	0	0
SVM	Standard Scaler	1	1	1	1
SVM	Sparse Normalizer	0	0	1	1
Random Forest	Standard Scaler	1	1	1	1
Kneighbors	Robust Scaler	0	0	1	1
Kneighbors	Sparse Normalizer	1	0	0	0
ExtraTrees	Min Max Scaler	0	0	0	0
# of classifiers used for ensemble algorithm		9	9	11	11

TABLE IV: Instances of classifiers and data transformation algorithms used for the Voting Ensemble algorithm for the keystroke inference experiments on *Under the Keyboard (UK), Plastic Surface (PS), Glass Surface (GS) and Metal Surface (MS)* placements. Each instance has a classifier and data transformation algorithm with different parameters.

Placement	Pre processing	Classifier	Parameters
UK	Max Abs Scaler	Light GBM	min leaf size = 20
PS	Standard Scaler	XGBoost	col sample: 0.5, eta: 0.05, max depth: 9, max leaves: 15, estimators: 400
GS	Max Abs Scaler	Light GBM	min leaf size = 20
MS	Max Abs Scaler	Light GBM	min leaf size = 20

TABLE V: Stand-alone classifiers used for the key inference attack for Under the Keyboard (UK), Plastic Surface (PS), Glass Surface (GS) and Metal Surface (MS) placements.

depicted by Figure 9. The figure attempts to answer the question of whether the distance of a key from the sensors has some relationship to how accurately it is inferred. For example 93.67% is the average accuracy of the keys 7, 8 and 9, while 90% is the average accuracy of the keys 0, 1, 4 and 7. As one moves from the top row towards the bottom row, average accuracy seems to reduce just as one moves from middle column to the left and right-most columns of numbers. These patterns suggest an increase in accuracy as one moves towards the sensors, however, this might need further investigations to see how strongly it holds up.

B. Detecting Keyboard Movements

Computer users do not move their keyboards that much given a permanently deployed computer system at work or home. However, movements indeed do happen, and, when the sensors are deployed under the keyboard, this might cause such an amount of misalignment between the keyboard and sensors that the attack degrades or even fails. We investigated

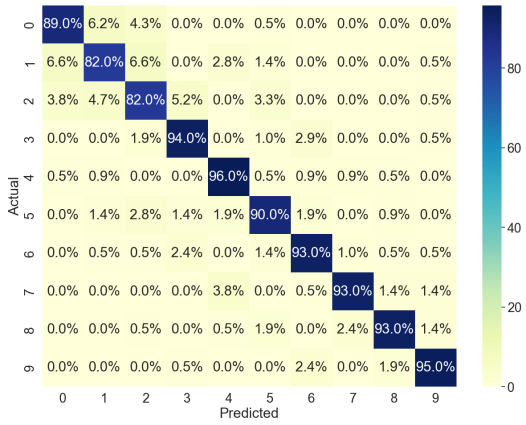


Fig. 8: Confusion Matrix for the Voting Ensemble when the sensor was under the keyboard

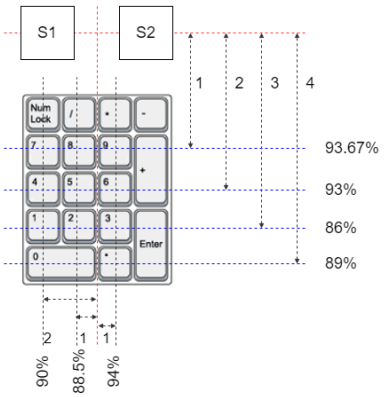
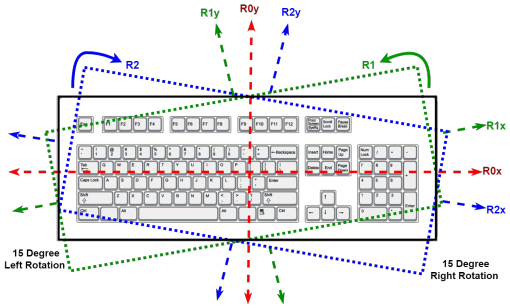
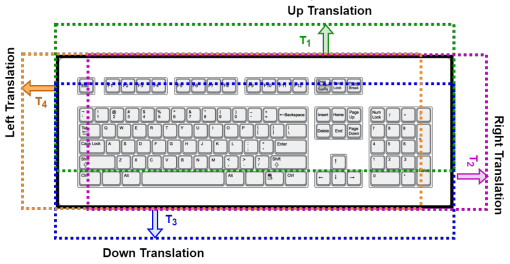


Fig. 9: Accuracy-Distance relation for the Voting Ensemble when the sensor was under the keyboard



(a) Rotation of keyboard



(b) Translation of keyboard

Fig. 10: Keyboard translations and rotations studied when the sensors were under the table. Rotations are through an angle of 15 degrees about the keyboard center while translations are 6cm in the directions indicated.

the impact of keyboard rotations and translations (see Figure 10) from the default position at which the attack was built. The rotations were 15 degrees clockwise and anticlockwise about the center of the keyboard (Figure 10a), while translations were 6cm in each of the four directions shown on Figure 10b.

Figure 11 shows a plot of 2 features for the rotation and translation cases when the sensors were under the plastic surface. We only discuss results from this surface as it reveals the general pattern seen with the other cases. Observe that in this 2D space, the cases are very well-clustered and quite distinct from the default positions, R0 and T0. This pattern suggests that a large catalogue of features coupled to a classification engine should easily detect these keyboard movements. When we proceeded and ran classification experiments we obtained classification accuracies of between 92.3% and 98.6%, confirming these keyboard movements should be easily detected.

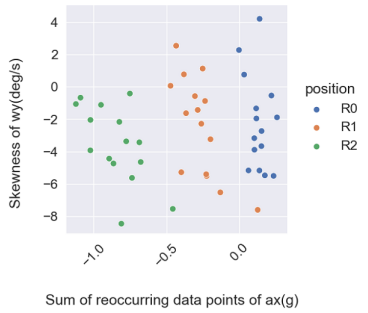
V. Related Research

In this section we discuss similar lines of research and how it differs from our own work.

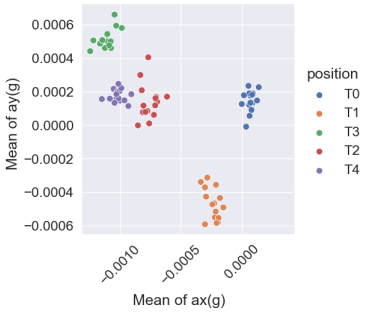
The study in [14] proposed a password extraction and inference system named Snoopy. They developed an app that listened to the smart watch's IMU sensors for any screen tapping or password swiping by the user. Using a deep neural network the attacker were able to detect passwords entered on Android or Apple watches.

Also, in [15] the authors studied an exploit that analyzed the data collected by the accelerometer at each screen tap or swipe event, a malicious users can, with a high accuracy, infer the input password/PIN or pattern. For every 5 guesses, the proposed model achieved 43% and 73% accuracy for PINs and patterns respectively in a controlled setting and 20% and 40% accuracy respectively, in uncontrolled scenario. In another study, [16] users' key-based PINs and/or passwords could be retrieved by analyzing data captured by motion sensors embedded in wearable devices. The study indicated that the patterns created by wrist movements carry enough information about which key the user typed on the keypad. The study in [17] investigated an attack aimed at inferring the words typed by a user while wearing a smart watch on their left hand, which provides substantial information for typed words inference.

This huge body of prior work emphasizes that motion sensor attacks pose a significant threat that is of much interest to the security community. However, these past works typically investigate scenarios where the keys typed on the virtual keyboards of smart-phones/watches. This puts our work apart from past research in at least the following three ways: (1) First, the small form factor of a phone/wearable should naturally result into a fundamentally different vibration mechanics relative to the case of the much larger computer keyboard where the keys are spread over a much larger area, (2) The



(a) Keyboard rotation



(b) Keyboard translation

Fig. 11: Impact of keyboard translation and rotation on the features captured by motion sensor. The meanings of rotations R0 through R2 and translations T0 through T4 are diagrammatically illustrated in Figure 10.

method of human interaction with a phone differs from that with a keyboard, indicating a different vibration dynamics between the two kinds of attacks (i.e., a phone is held in one hand while the other hand types while a computer keyboard is placed on a table while the hand(s) type on it). The above 2 points imply that past research which showed these attacks on a phone/watch would not necessarily answer the question of whether the same attacks are feasible on a computer keyboard, (3) Attacks in the smart phone/watch scenario assume a malware to somehow be installed on the phone/watch so as to record the vibrations and send the data to the attacker. This is a very strong assumption that we do not make in our attack as we simply require the attacker to attach Bluetooth-enabled IoT sensors to the target.

To summarize, the idea that multiple *sensors could be glued under a desktop keyboard and successfully capture vibration patterns unique to keys on the keyboard* is a new line of attack that no past research has explored. Further, the question that a series of *motion sensors hidden under a table could leak data typed on a keyboard located on the top of the table* has never been studied before. Our work is the first to examine these lines of attack.

VI. Conclusion

In this paper, we have designed and evaluated an attack in which motion sensors hidden under the computer keyboard or table on which a keyboard sits are used to spy on keyboard inputs. We have shown the attack to be highly effective when the sensors are hidden under the keyboard, and somewhat less effective though still posing a significant threat when the sensors are under plastic, metallic or glass tables. This is a highly practical attack given that the sensors required are cheaply available on the market, come packaged with all the supporting software and thus require no hardware knowledge, and most importantly, do not require the attacker to circumvent any defenses on the victim's computer.

References

[1] W. motion, "Wt901blecl mpu9250 high-precision 9-axis gyroscope+angle(xy 0.05° accuracy)+magnetometer with kalman filter, low-power 3-axis ahrs imu sensor for arduino," https://www.wit-motion.com/9-axis/wt901blecl-mpu9250-high-precision.html, (accessed January 31, 2022).

[2] Logitech, "Mk270 wireless keyboard," https://www.logitech.com/en-us/products/combos/mk270-wireless-keyboard-mouse.920-004536.html, (accessed January 31, 2022).

[3] "Number generator," https://numbergenerator.org/random-4-digit-number-generator#!numbers=5&length=8&addfilters=, (accessed January 31, 2022).

[4] M. Christ, "tsfresh.feature-extraction package," https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html, 2016 (accessed January 5, 2022).

[5] ——, "tsfresh.feature-selection package," https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html, 2016 (accessed January 5, 2022).

[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[7] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *NIPS*, 2017.

[8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[9] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr 2006. [Online]. Available: https://doi.org/10.1007/s10994-006-6226-1

[10] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, R. Meersman, Z. Tari, and D. C. Schmidt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 986–996.

[11] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[12] scikit-learn developers, "Preprocessing data," https://scikit-learn.org/stable/modules/preprocessing.html, 2007 (accessed January 10, 2022).

[13] A. Dogan and D. Birant, "A weighted majority voting ensemble approach for classification," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 2019, pp. 1–6.

[14] C. X. Lu, B. Du, H. Wen, S. Wang, A. Markham, I. Martinovic, Y. Shen, and N. Trigoni, "Snoopy: Sniffing your smartwatch passwords via deep sequence learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–29, 2018.

[15] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of accelerometer side channels on smartphones," in *Proceedings of the 28th annual computer security applications conference*, 2012, pp. 41–50.

[16] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu, "Friend or foe? your wearable devices reveal your personal pin," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016, pp. 189–200.

[17] H. Wang, T. T.-T. Lai, and R. Roy Choudhury, "Mole: Motion leaks through smartwatch sensors," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 155–166.