smartphone Side-Channel Attacks and Defenses

**Module 5 lAB Manual**

**Lab Manual Development Institution:** Colorado School of Mines

**Lab Manual Contributors**:

Weiping Pei\*, Zhiju Yang\*, Chuan Yue@

(\*: PhD student and research/teaching assistant; @: Faculty.)

# Motion sensor data perturbation for defense

**Lab Description:** In this lab, you will design defense techniques to protect against motion sensor based side-channel attacks. In more details, you will experiment with some representative data perturbation techniques (such as reducing data sampling frequency and adding noises to the data) that can help reduce the quality of the motion sensor data for defending against cross-site input inference and user fingerprinting side-channel attacks. You will describe and justify the basic idea and intuition of your defense techniques. You will evaluate the effectiveness of such defense techniques, for example by comparing the accuracy of the attacks between with and without using these defense techniques. You will discuss the potential side-effects of these defense techniques. We suggest you to reuse your code in Labs 3 and 4 to perform this lab. This lab consists of six STEPs.

The high-level **learning outcomes** and the corresponding **assessment** of this lab are summarized as follows. In other words, upon completion of this lab, students should be able to:

* **Interpret** two representative data perturbation based defense techniques (reducing data sampling frequency and adding noises to the data).
  + Assessed by the tasks and outputs specified in STEP 1.
* **Construct** new datasets by applying two data perturbation based defense techniques.
  + Assessed by the tasks and outputs specified in STEP 2.
* **Create** machine learning models with training, hyper-parameters tuning, and evaluation for performing input inference attacks using the newly constructed datasets.
  + Assessed by the tasks and outputs specified in STEP 3.
* **Create** machine learning models with training, hyper-parameters tuning, and evaluation for performing user fingerprinting attacks using the newly constructed datasets.
  + Assessed by the tasks and outputs specified in STEP 4.
* **Compare** two data perturbation based defense techniques in terms of their effectiveness on reducing the attack accuracy.
  + Assessed by the tasks and outputs specified in STEPs 3 and 4.
* **Estimate** the potential side-effects of two data perturbation based defense techniques.
  + Assessed by the tasks and outputs specified in STEP 5.
* **Propose** new ideas to protect against motion sensor based side-channel attacks.
  + Assessed by the tasks and outputs specified in STEP 6.

**Lab Environment:** Linux, Mac, or Windows.

**Lab Files that are Needed:** TheLab Manual file and the motion\_data.pkl file.

**Learning Setting:** This lab module is for students to complete outside the classroom, so it can be used in either face to face or online courses.

**Prerequisites:** Java or Python Programming, Basic Cybersecurity and Machine Learning knowledge and skills, Linux or Windows Systems, Computer Networks.

**Length of Completion:** 600 minutes.

**Level of Instruction:** Senior undergraduate students or graduate students in CS or related STEM programs. The lab exercise should be further simplified if it will be used for freshmen, sophomores, or none-CS major students.

**Interconnection with Other Labs:** This lab module is standalone by itself; however, if needed, an instructor can use the details in the course project manual and the other four lab manuals to provide additional hints to students.

**Assessment Guideline:** Students should follow the steps to answer all the questions. Based on the points assigned to each individual question, the instructor will grade each answer (together with the additional materials if specified for the question) in terms of its correctness (60%), clarity (20%), and concision (20%).

### **Lab Exercise/step 1 (interpret two representative data perturbation based defense techniques)**

There are many ways to defend against motion sensor based side-channel attacks, more specifically, cross-site input inference attacks and user fingerprinting attacks. One extreme is to completely block webpages’ access to the motion sensor data, but this approach will immediately nullify the benefits of using motion sensor data in HTML5 for richer functionality and better interactivity. The other extreme is to always ask a user to grant or deny motion sensor data access requests on individual webpages, but this approach will not be usable or effective because users often do not pay attention to or do not understand permissions and often become habituated to granting permissions. Therefore, it is important to design fine-grained defense approaches that could be more usable and effective in practice.

One usable and effective approach is to perturb the motion sensor data that could be collected by JavaScript on malicious webpages. Two representative data perturbation techniques are: reducing data sampling frequency, and adding noises to the data.

**Question 1**: The sampling frequency in the original motion sensor data (CSM\_MotionSensor\_Dataset.zip) and the preprocessed motion sensor data (motion\_data.pkl) is 60Hz, i.e., you roughly have 16~17 data points for each type of data in each second. If three smartphones provide the motion sensor data at the 30Hz, 20Hz, and 15Hz frequency, respectively, how many data points can you roughly obtain in each second from the three smartphones, respectively? Which data file(s) that you have encountered so far in the previous labs (i.e., CSM\_MotionSensor\_Dataset.zip, motion\_data.pkl, or feature\_data.csv) could be good for you to directly derive such three low-frequency datasets? Please explain your answers.  
(Total score: 5 points. Grading rubric:  
100% points for correctly answering both questions with explanations;  
60% points for correctly answering one of the two questions with explanation;  
30% points for correctly answering one of the two questions without explanation.)

**Question 2**: The nine types of data (the acceleration forces in the x, y, and z directions, the rotation rates in the alpha, beta, and gamma directions, and the acceleration forces with gravity in the x, y, and z directions, respectively) in either CSM\_MotionSensor\_Dataset.zip or motion\_data.pkl contain the raw values. If we want to add some ransom noise to them without making them completely useless (e.g., without changing all of them to some constant values), what could be some ideas and why? Which file(s) (CSM\_MotionSensor\_Dataset.zip, motion\_data.pkl, or feature\_data.csv) could be good for you to directly derive such noise-added datasets? Please explain your answers.  
(Total score: 5 points. Grading rubric:  
100% points for correctly answering both questions with explanations;  
60% points for correctly answering one of the two questions with explanation;  
30% points for correctly answering one of the two questions without explanation.)

**Question 3**: Why these two data perturbation techniques could be helpful for you to defend against input inference attacks and user fingerprinting attacks?  
(Total score: 5 points. Grading rubric:  
100% points for a clear explanation;  
50% points for a vague explanation.)

### **LAB EXERCISE/STEP 2 (construct six new quality-reduced datasets)**

Download the file motion\_data.pkl for the preprocessed motion sensor data that was produced in Lab 2, and write code to load the data, e.g., by using the Python *pickle* module. This should result in a long list of 3-tuples in your code for the motion sensor data of 30 users. Recall that the first element of each 3-tuple is a matrix with 9 rows where each row is one of the 9 sequences of sensor data, the second element is the key or digit (e.g., digit “5”) typed by a user, and the third element is the ID of the user who typed the key. Each 3-tuple has the following format:

([ (accX\_1, accX\_2, …),

(accY\_1, accY\_2, …),

…,

(accgZ\_1, accgZ\_1, ...)],

“5”,

UserID)

Reviewing the manuals for Lab 2 and Lab 3 will be very helpful for you to perform the tasks in this step. In Lab 2, the input is the file motion\_data.pkl, and the output is the CSV (comma-separated values) feature data file named feature\_data.csv. Now, reuse your code in Lab 2 and add something new to reduce the quality of the original data, and then generate the following six feature data files: data\_freq\_2.csv, data\_freq\_3.csv, data\_freq\_4.csv, data\_noise\_0.2.csv, data\_noise\_0.5.csv, and data\_noise\_0.8.csv.

The first three feature data files should be generated by reducing the data sampling frequency. Basically, you first extract, from each sequence of the original 60Hz data, the first of every two, three, and four data points to derive three new low-frequency datasets with the sampling frequencies of 30Hz, 20Hz, and 15Hz, respectively. You then extract the statistical features as in Lab 3 to generate the following three feature data files:

* + **data\_freq\_2.csv**: generated by reducing the data sampling frequency from the original 60Hz (as in motion\_data.pkl) to 30Hz.
  + **data\_freq\_3.csv**: generated by reducing the data sampling frequency from the original 60Hz (as in motion\_data.pkl) to 20Hz.
  + **data\_freq\_4.csv**: generated by reducing the data sampling frequency from the original 60Hz (as in motion\_data.pkl) to 15Hz.

The next three feature data files should be generated by first adding noise to the data, and then extract the statistical features as in Lab 3. There could be many ways to add the noise. We require you to use the following formula to add the noise:

newvalue = oldvalue + max\_value \* ratio \* random();

Here, *max\_value* is the maximum value of one type of data for a specific digit typing, e.g., it is the maximum value in (accX\_1, accX\_2, …) or in (accY\_1, accY\_2, …) for a digit typing. The function *random()* should return a real value between 0 and 1 inclusively. The *ratio* should be 0.2, 0.5, or 0.8 in this lab. That is, each accX\_i in a row will be added with “*max (accX\_1, accX\_2, …) \* ratio \* random()*” and updated to a new value, each accY\_i in a row will be added with “*max (accY\_1, accY\_2, …) \* ratio \* random()*” and updated to a new value, and so on for all the nine types of data in all the 3-tuples. Correspondingly, you will generate the following three feature data files;

* + **data\_noise\_0.2.csv**: generated by using ratio=0.2 in the formula.
  + **data\_noise\_0.5.csv**: generated by using ratio=0.5 in the formula.
  + **data\_noise\_0.8.csv**: generated by using ratio=0.8 in the formula.

**Question 4**: What is your code for constructing these six new feature data files?   
(Total score: 15 points. Grading rubric:  
100% points for correct and complete code;  
60% points for partially correct and partially complete code;  
30% points for partially correct or partially complete code.)

### **LAB EXERCISE/STEP 3 (create machine learning models for performing input inference attacks)**

In this step, you will use each of the six newly constructed datasets (i.e., feature data files) to perform the input inference attacks. Similar to the STEPS 1~5 described in Lab 4, ***for each of the six new datasets***, you will (1) construct the data structures “X” (the feature vectors) and “y” (the list of labels), (2) split the data into training and testing data, (3) train, tune, and evaluate two models (i.e., using two different algorithms) to perform input inference attacks, and (4) analyze the evaluation results.

**Question 5**: What are the two algorithms or models you used? What hyper-parameters and their values worked best for your models? What accuracy did your two models achieve on the testing data? (Hint: You can use GridSearchCV method in scikit-learn to tune your models). Note that since you have six newly constructed datasets and you use two different algorithms, you will have 12 sets of new experimental results. You should also include the two sets of experimental results based on the unperturbed dataset for the two algorithms.  
(Total score: 15 points. Grading rubric:  
100% points for answering all the three questions and providing the 14 sets of experimental results;  
60% points for answering two of the three questions with incomplete experimental results;  
30% points for answering one of the three questions with incomplete experimental results.)

**Question 6**: Analyze these 14 sets of experimental results for input inference attacks. What are your observations when you compare the results between the two types of data perturbation techniques, among different datasets constructed with different parameters, and between the two algorithms?   
(Total score: 10 points. Grading rubric:  
100% points for providing the observations and comparisons from all the three aspects;  
60% points for providing the observations and comparisons from two of the three aspects;  
30% points for providing the observations and comparisons from one of the three aspects.)

### **LAB EXERCISE/STEP 4 (create machine learning models for performing user fingerprinting attacks)**

In this step, you will use each of the six newly constructed datasets to perform the user fingerprinting attacks. Similar to what you did in STEP 3 of this lab but with the goal similar to that of STEP 8 described in Lab 4, ***for each of the six new datasets***, you will (1) use the “user\_id” column as the label in the list “y” instead of the “key\_pressed” column, (2) split the data into training and testing data, (3) train, tune, and evaluate two models (i.e., using two different algorithms) to perform user fingerprinting attacks, and (4) analyze the evaluation results.

**Question 7**: What are the two algorithms or models you used? What hyper-parameters and their values worked best for your models? What accuracy did your two models achieve on the testing data? (Hint: You can use GridSearchCV method in scikit-learn to tune your models). Note that since you have six newly constructed datasets and you use two different algorithms, you will have 12 sets of experimental results. You should also include the two sets of experimental results based on the unperturbed dataset for the two algorithms.  
(Total score: 15 points. Grading rubric:  
100% points for answering all the three questions and providing the 14 sets of experimental results;  
60% points for answering two of the three questions with incomplete experimental results;  
30% points for answering one of the three questions with incomplete experimental results.)

**Question 8**: Analyze these 14 sets of experimental results for user fingerprinting attacks. What are your observations when you compare the results between the two types of data perturbation techniques, among different datasets constructed with different parameters, and between the two algorithms?  
(Total score: 10 points. Grading rubric:  
100% points for providing the observations and comparisons from all the three aspects;  
60% points for providing the observations and comparisons from two of the three aspects;  
30% points for providing the observations and comparisons from one of the three aspects.)

### **LAB EXERCISE/STEP 5 (estimate the side-effects of the two data perturbation defense techniques)**

You might have already observed that either reducing the sampling frequency or adding noises, the input inference accuracy as well as the user fingerprinting accuracy can be reduced. This is desirable from the security and privacy protection perspective. However, an important question is that: while positively reducing the attack accuracy, whether these two data perturbation techniques also negatively compromise the utility of motion sensor data and affect the functionality of legitimate Web or mobile applications.

Considering two application scenarios where coarse-grained motion data are still needed for basic functions. In one scenario, some applications are only interested in detecting if some tapping-like activities are performed in a certain period of time; in the other scenario, applications are interested in quantifying the number of tappings performed in a certain period of time. In other words, these applications do not need to know which keys are tapped and who tapped those keys.

**Question 9**: Will the two data perturbation techniques that you have experimented with still allow those applications to perform some coarse-grained functions (e.g., detecting tapping-like activities or quantifying the number of tappings)? Considering the side-effects of these two techniques, do you recommend Web browser or smartphone vendors to deploy such defense techniques? Please explain your answers. (Hint: you may want to answer these questions either based on your results analyses in STEPs 3 and 4, or based on some new analyses such as plotting the perturbed data to visually inspect them. A visual inspection example is there in the Figure 14 “A representative example of perturbing z axis acceleration force data of some letter inputs in 15 seconds” of the IEEE TIFS 2019 paper titled “[Sensor-Based Mobile Web Cross-Site Input Inference Attacks and Defenses](https://ieeexplore.ieee.org/document/8371301/)”; in your case, you only need to plot the data points for individual tappings.)  
(Total score: 10 points. Grading rubric:  
100% points for correctly answering both questions with explanations;  
60% points for correctly answering one of the two questions with explanation;  
30% points for correctly answering one of the two questions without explanation.)

### **LAB EXERCISE/STEP 6 (Propose new ideas to protect against motion sensor based side-channel attacks)**

As described in STEP 1, it is important to design fine-grained defense approaches that could be more usable and effective in practice; meanwhile, perturbing the sensor data that could be collected by JavaScript on malicious webpages is just one of the fine-grained defense approaches. Also as you analyzed in STEP 5, the data perturbation approach does have some limitations, for example, in terms of the side-effects.

**Question 10**: Propose two concrete ideas on how other fine-grained approaches may help defend against input inference attacks and user fingerprinting attacks. Explain these approaches by comparing them with the data perturbation approach. (Hint: you may consider the solution and the corresponding comparison from multiple perspectives such as the development effort, deployment effort, effectiveness in reducing the accuracy of the attacks, and the usability.)  
(Total score: 10 points. Grading rubric:  
100% points for proposing two concrete ideas with the clear explanations;  
60% points for proposing one concrete idea with the clear explanation;  
30% points for proposing one concrete idea without the clear explanation.)

### **Puzzler (N/A)**

This is an advanced activity for students who complete the regular activities early. N/A for this lab.

## What to submit

Please answer all the 10 questions in this lab exercise. Please feel free to directly reuse this Word document to provide and submit your answers. Please submit your complete code for performing all the dataset construction, model training, tuning, and evaluation steps in this lab.