5.3.2018

Acoustic Side-Channel Attack:

Distinguishing Encryption from Decryption Using an iPhone

Aakash Patel

Steven Sedig

5.3.2018

Project Summary

We started this project with the huge challenge of trying to extract an RSA key via an acoustic side-channel attack.

In the beginning steps of the project, we used an iPhone as a microphone and used audio analysis tools to see if we could visually differentiate between encryption and decryption. Once we were able to confirm visual differences, we moved on to recording audio samples. We tried using another software tool to see if we could differentiate and overlay the audio samples; this worked to a certain extent, but it didn't give us the greatest results, so we tried finding other tools which would yield better results.

We came upon the GitHub repository of Theodoros Giannakopoulos, a Post-Doc research associate at Demokritos in Greece. This repository helped us out a lot, because Theodoros had created an audio classifier in Python which we were able to train to differentiate between encryption and decryption.

Recording Setup

Our recording setup consisted of a target laptop, a microphone, and a recording laptop.

The target laptop was a Lenovo T61 running GnuPG 2.2.6. We removed the keyboard to guarantee excellent acoustic emissions, but in principle this would be unnecessary with a sufficiently well-trained acoustic model.

The recording device was an iPhone 6s Plus, and the recording app was Megaphone. The Megaphone app allowed us to choose which microphone to use and allowed us to send live recording data to the recording laptop.

The recording laptop was a Windows 10 machine running a generic audio recording software and a real-time audio analysis software called Friture 0.35.

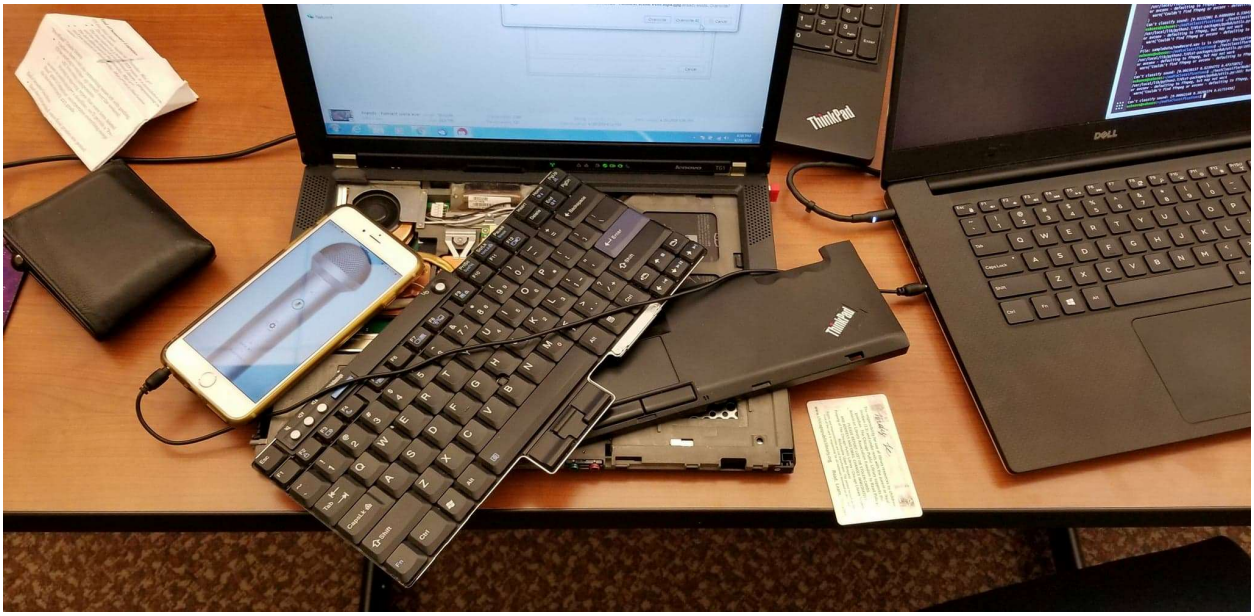

Figure 1: Recording Setup

Encryption and Decryption Recording

We encrypted and decrypted three types of files using GnuPG: a large application, a medium-sized mp4, and a small mp4.

During the encryption and decryption processes, we recorded audio via the iPhone microphone placed close to the T61's processor. Initially, we used Friture to visualize the frequency distribution of the audio. We noted that there was a clear difference between the idle processor state and an encryption or decryption operation (Figure 2, Figure 3). This gave us the confidence to use an audio classifier on the recorded data.

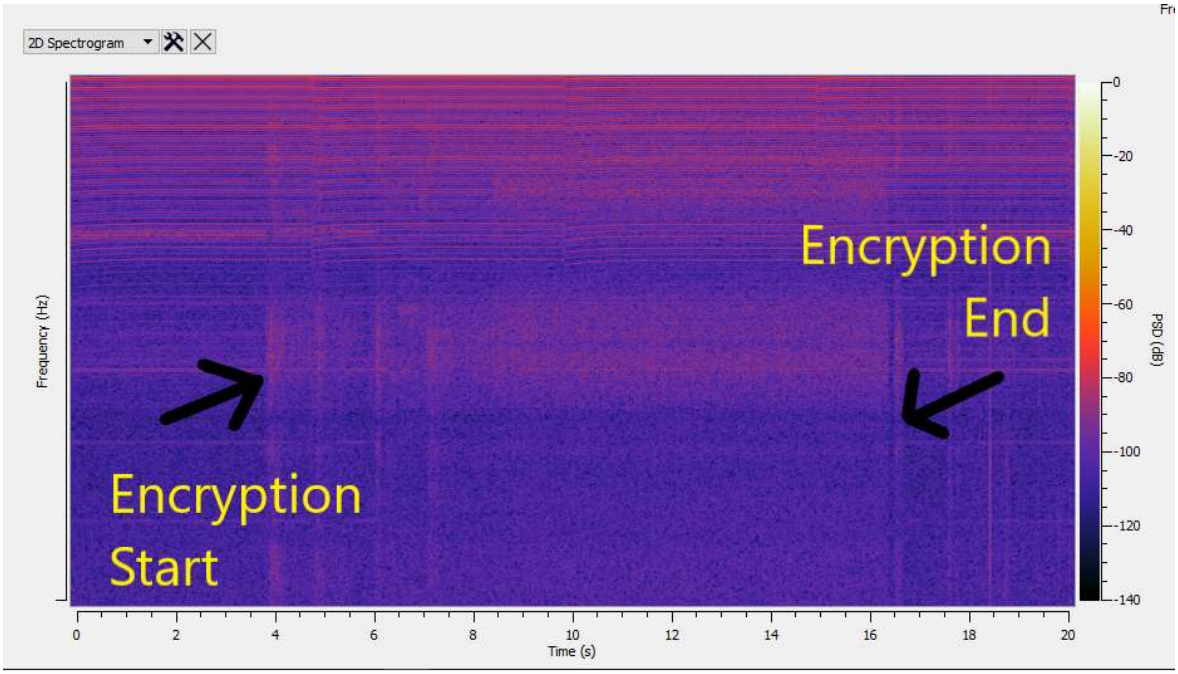

Figure 2: Encryption

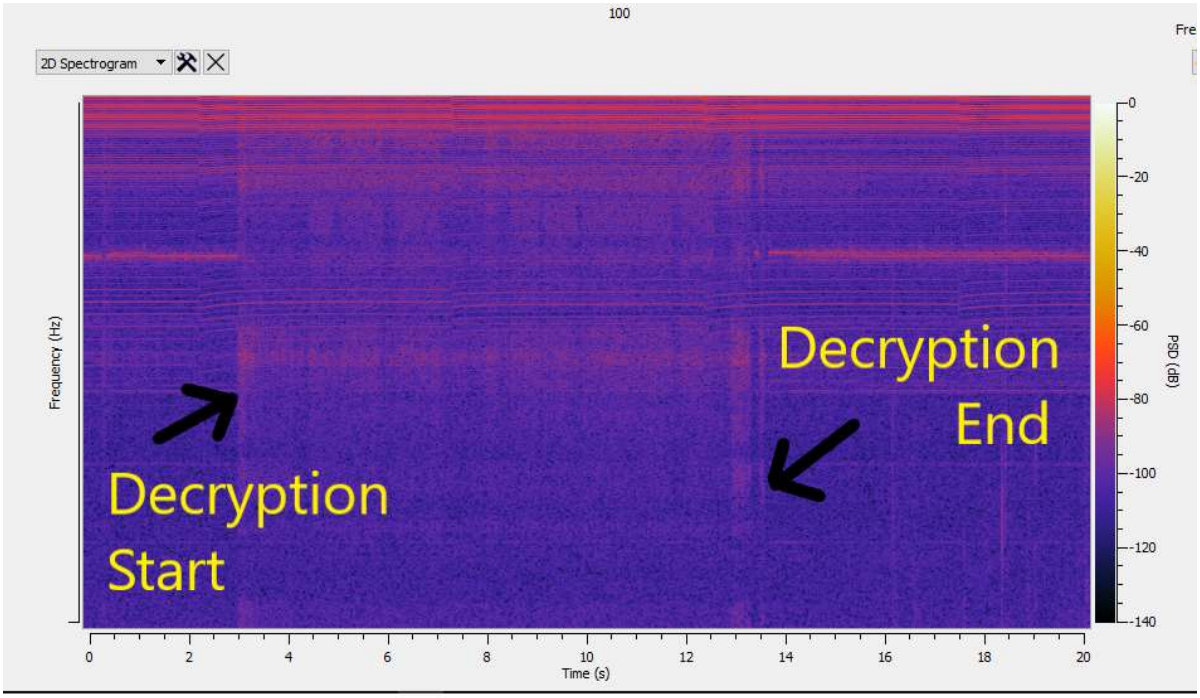

Figure 3: Decryption

Results

After obtaining our encryption and decryption audio, we trained an audio classifier (described in 'Project Summary' and 'How to Demo Our Project') using the data. The results were quite significant: in most of the audio samples we tested our classifier on, the predicted processor state matched the actual processor state with a high probability.

How to Demo Our Project

Clone our Git Repository (on Linux): https://github.com/DrFicus/CS460_Final_Project

There are two folders in our final project demo code.

The first folder, *FinalProjectResearch_Control*, contains data that we collected with very few variables. For example, we ensured that the CPU fan was always running, we stopped recording immediately after encryption or decryption completed, and we used the same file each time.

The second folder, *FinalProjectResearch_Experimental*, contains data that we collected with more variables. We didn't worry about the CPU fan being on or off, we randomized recording time before and after encryption or decryption, and we used different files each time.

Change your directory to:

FinalProjectResearch_Control/FinalProjectResearch/audioClassifications

The *audioClassifications* directory contains several files and folders: *createClassifierModel*, *testClassifierModel*, *trainingData*, and *sampleData*.

The *trainingData* directory has three sub-directories: *EncryptionData*, *DecryptionData*, and *RegularProcessingData*. The *EncryptionData* sub-directory contains audio recordings from when we encrypted files, the *DecryptionData* sub-directory contains audio recordings from when we decrypted files, and the *RegularProcessingData* sub-directory contains audio recordings from when we just had Google Chrome running.

These three sub-directories were used to train the audio classifier. We created a small Python script to train the audio classifier and this script is called *createClassifierModel*. To train the model, run the following command via the command-line:

$ ./createClassifierModel trainingData

We also have a directory called *sampleData* which contains five audio recordings of either decryption or encryption. These were recorded separately from the *trainingData* audio files. To test if an audio file is encryption, decryption, or just normal processing, run the following command via the command-line, replacing *'X'* with a number between 1 and 5:

$ ./testClassifierModel sampleData/newRecordX.wav

The output of the command will either specify which label was chosen for the audio file or it will just say '*Sound classification not certain*'. We also took the liberty to print out the actual percentages that the model generated. While testing the model we sometimes found that the audio file would not get classified, but it would evaluate one label higher than others. Looking at those evaluations we confirmed that the software would correctly classify the audio if the probability was set to a lower percentage. We set the probability at 0.7, which means that any label with a probability greater than or equal to 0.7 will get chosen as the classified label for the audio file.

Repeat the same exact process to run and test the demo for the Experimental data in the *FinalProjectResearch_Experimental* folder. This data includes relevant audio files of encryption and decryption, but with random durations and random start and end times as mentioned before in step 1.

Reflections on Previous Work

Initially, the goal of this project was to demonstrate an acoustic side-channel attack on RSA using an Amazon Echo (see the last page of this document). However, after hours of research, we learned that Amazon has done a fairly good job of making the raw Echo audio data unavailable to developers. So, failing the development of an exploit for the Echo during the semester, we were stuck. Therefore, we decided to develop our attack in a similar manner to that already done by others (Genkin).

After deciding to use the methods provided in Daniel Genkin's paper, we searched for methods of recording ultrasound. Ultrasonic emissions from CPUs are known to emit useful information about the state of the processor and can be used to decipher RSA keys. However, despite the potential usefulness of such a device, we were unable to locate one at an affordable price. Some thought was given to developing our own hardware, but time constraints and development costs scrapped this idea.

These steps led us to develop the successful methods already described in this paper.

5.3.2018

New Findings

While recording audio files for this project, sometimes we would be typing on the recording laptop and our microphone (iPhone) was able to pick up this sound. Theoretically, instead of feeding the audio classifier audio files of encryption and decryption data, we could have fed it audio files of each key on a keyboard being pressed. We might then have been able to create an audio classifier model to differentiate keypresses.

Additional Thoughts on Side-Channel Attack

Side-Channel attacks are definitely a threat. However, after completing this project we have realized that many variables need to be controlled to perform a side-channel attack. For example, if we switched out the processor we were listening to with a processor with better processing speeds, then the audio classifier we trained would need to be trained again. In a real-life scenario with an RSA side-channel attack using acoustics, one would need access to the actual laptop or desktop they are trying to get the RSA key from.

Additionally, processors are made from silicone and usually there are variations in how lithography is done. This means that every batch of processors will have slight differences and recording audio from these processors with ultrasonic microphones might reveal these differences. What this translates to for a side-channel attack is that even if your target was using a processor such as an Intel i7-7700K, or an AMD Ryzen 7 1800X, and you obtained the same exact processor to train your model, your model still might not help you get the RSA key because the minor differences in processors could lead to huge differences in what the ultrasonic microphone picks up.

5.3.2018

(Original) CS 460 Project Proposal

Outline

For our project in CS 460 this semester, we plan to demonstrate a side-channel attack on RSA using an Amazon Echo. We are basing our demonstration on a paper in which researchers were able to decode a 4096-bit RSA key within one hour by using a smartphone microphone placed near a laptop performing RSA signing (or decryption) operations (Genkin).

Deliverables

We will demonstrate that our code is able to determine the difference between a zero and a one of an RSA key in use by a laptop running a program of our choosing near an Amazon Echo.

Works Cited

Genkin, Daniel, et al. "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis." *Advances in Cryptology – CRYPTO 2014 Lecture Notes in Computer Science*, 2014, pp. 444–461., doi:10.1007/978-3-662-44371-2_25.