Bachelor's Thesis Computing Science

# Stealth Acoustic Communication Application

Jelle Medendorp
s1027748

June 23, 2023

*First supervisor/assessor:*
prof. dr. Lejla Batina

*Second assessor:*
prof. dr. Martha Larson

Radboud University

**abstract**

A lot of sensitive information sensors inside common smartphones have been secured properly and can only be used with the users permission. But some sensors seen as not sensitive, like the gyroscope, can still be used for permission-less side channel attacks. These side channel attacks can be used to communicate vulnerable data between other devices that are being used in close proximity. In this thesis I have done a thorough literature review and built an application that is able to stealthy receive binary through the form of acoustic waves. Showing that, phone manufactures need to reconsider if access to these type of sensors should be this unrestricted.

# Contents

This thesis is centered around the development of an IOS application. The source code of this application can be found anytime at:

`https://github.com/JelleMedendorp/TrackingApp`

More info on running the application can be found in the appendix.

# Chapter 1

# Introduction

In today's day and age, almost everyone walks around with a mobile phone in their pocket. The massive amount of capabilities this device gives us comes accompanied with a lot of security risks. Luckily, through the years a lot of research has gone into the security of mobile phones, and most of the vulnerabilities have been remediated, making most mobile phones fairly secure devices. Sensitive information sensors, like the microphone, the camera, and the phone's location, all require explicit permission from the user to be accessed, and thus are fairly secure. However, recent research shows that maybe this is not enough[1], and that some sensors which are commonly seen as not sensitive, and thus can be used without explicit permission, can be used to leak vulnerable information while remaining stealth.

In this thesis we are going to focus on the phone's gyroscope sensors. In particular MEMS (Micro Electronic Mechanical System) Gyroscopes, as these are used in almost all mobile phones. A gyroscope is used to sense rotation of the device. At first glance, it looks fairly simple, and not prone to security risks. But research has shown us otherwise[8]. We will use the fact that gyroscope readings can be tampered through audio, and apply the existing theory to create an application for mobile phones, which can receive information from devices in close proximity, without permission and knowledge of the user.

In the next chapter we will explain the necessary preliminaries needed for understanding the theory. In chapter 3 we will discuss the relevant work that was used as guidance for this thesis. Chapter 4 covers some experiments we have done to help with the implementation. Chapter 5 will be about the implementation itself. We have conducted some test cases in chapter 6. In chapter 7 we have a discussion about the outcomes of this thesis and conclude it. And chapter 8 will round it off by looking ahead for some possible future work.

# Chapter 2

# Preliminaries

Before we dive into the relevant work and the implementation of the application, we will briefly provide some definitions and background information about the MEMS gyroscope sensor and the setup used in our experiments and application.

## 2.1 MEMS Gyroscope

Micro Electro Mechanical Systems (MEMS) are used in all kinds of different fields, like vehicles, gaming consoles, compasses, mobile phones, and many more. They are widely used due to their low-power consumption, small size, and easy fabrication in large quantities[8]. A MEMS Gyroscope gives us information about the orientation of a device, in particular the rotation around its own x, y, and z axes. This works on the premise of the Coriolis Effect[13]. Which shows us that when an object is moving along the x-axis, and an angular effect occurs on this object, the mass will also displace on the y-axis. By having a sensing mass attached to strings move back and forth on a certain axis, measuring displacement perpendicular to this axis allows the device to calculate the amount of rotation that occurs on this axis. Thus giving us information about the orientation of the device. An schematic overview of this structure can be found in figure 2.1.

### 2.1.1 Resonance

The sensing mass is continuously vibrating along an axis at a certain frequency, the frequency of this vibration is called the resonance frequency[10]. Previous research has shown us that it is possible for the movement of this sensing mass to be altered by acoustic waves, making the sensing mass also vibrate in the Coriolis force axis, disturbing the readings[4][8]. For the sensing mass to be influenced by acoustic waves in a constant way, we need to have a frequency that is equal to or very close to the resonance frequency of

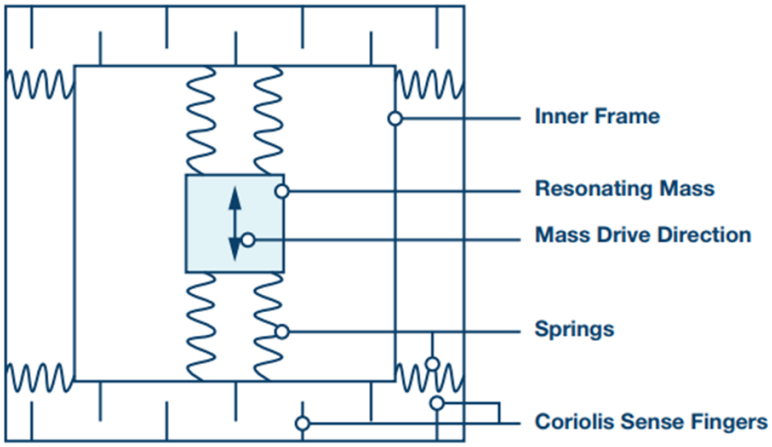



Figure 2.1: Schematic of the gyroscope's mechanical structure [16]

the MEMS Gyroscope. This resonance frequency, which is easily obtained through simple trial and error tests, is often between 19 and 30kHz. Since humans can only hear sounds between 20Hz and 20kHz[5], with the higher frequencies degrading the older someone gets, it leaves room for acoustic attacks that can go unnoticed by the human ear.

## 2.2 Oscillator

For generating the required acoustic waves, we will be using an oscillator. We need the acoustic waves to have a very precise frequency and timing since we want to be able to turn it into data. An oscillator can do just that for us. For this thesis, we used the Rigol DG822, 2 channel oscillator. This made us able to send waves of a precise frequency, at a precise interval. With this, we can differentiate between interval lengths and map different intervals to different binary sequences, which in the end allows us to send a multitude of different binary sequences. We go deeper into the way we turned the acoustic waves into data in chapter 5. On the right you can see a picture of the oscillator that was used.

Figure 2.2: Picture of the used Rigol DG822, 2 channel oscillator

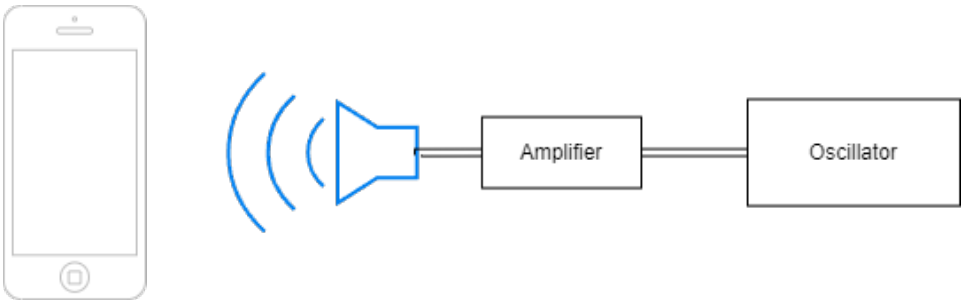



Figure 2.3: Schematic overview of the testing setup

## 2.3 Setup

To have a working setup, just the oscillator is not enough, since the oscillator does not have a built-in speaker. We connected the output of the oscillator to an amplifier. In this case, the Lepy Stereo Class-D Digital Audio Amplifier. This is then connected to a speaker, which we can aim at the phone. On said phone, we run the application that is created for this thesis, which allows us to see the gyroscopic data, and then we try to map it back to binary. In figure 2.3 one can see a schematic overview of the used setup

# Chapter 3

# Related Work

To get a better understanding of how and why this research topic arose, we will discuss some relevant research that has been done in the past. We discuss relevant research works that studied existing vulnerabilities in the MEMS sensors and research works that this thesis builds further upon.

Like explained in the preliminaries, MEMS gyroscopes use displacement as a way to measure rotation on all different axes. Previous works shown us that MEMS gyroscopes do not only respond to the movements of the device itself, but are also susceptible to external interference[8]. This can be done via acoustic injection attacks [15], by injecting acoustic waves close to the resonant frequency of the given device. Or it can be done via vibrations[17]. Both of these work on the concept that the gyroscope is constantly vibrating, and other vibrations can alter this movement, distorting the readings of said device.

The MEMS Sensors are also used to navigate drones. Yunmok Son et al. [13] shown us that by injecting acoustic signals with a specific frequency, the measurements read from the sensor can be altered, which can lead to miss-navigation or the crash of the drone under attack.

Zhen Hong et al. [7] shown how an Electronic Stability Program (ESP), which is widely used in modern vehicles, can also be spoofed via acoustic waves. They also developed a method of hiding these malicious signals in music one might play in their car, altering the wave forms just so slightly that a human ear usually cannot distinguish them.

Reading the MEMS sensors data can also lead to privacy issues. Yan Michalevsky et al. [11] found that MEMS gyroscopes used on modern smartphones are sufficiently sensitive to measure acoustic waves, such as conversations, in close vicinity of the phone. Which can reveal private information about the phone's environment, like who is speaking and, to some extend, what is being said.

Another vulnerability of reading the MEMS sensors data, is the fact that typing on a smartphone with a touchscreen, which is most smartphones

these days, will cause different motions of the phone depending on what key is pressed [3][12][2]. This, while the gyroscope can be freely used without the need for permission, could give us information about private information such as PIN codes or bank account logins.

Jan Han et al. [6] demonstrated how the MEMS accelerometer, which is used to calculate the acceleration of a device, can be used to infer the trajectory and starting point of an individual who is driving. Thus being able to locate this person. Another way of tracking devices is discussed by Nikolay Matyunin et al. [10]. In this paper, they describe how injecting information from acoustic waves and reading it without permission through a device's gyroscope could reveal the location of a device.

The theory is that when you have an application reading the gyroscope data, that is able to differentiate between different acoustic disturbances it picks up. Accompanied by playing these different disturbing acoustic waves at different locations. The application will know at which one of these different locations the device is located once it picks up one of these acoustic waves. Thus revealing the location of the device without the need for permission. This paper describes the theories and ideas for how this could work. We have taken these and made them into a real-life, usable implementation, with a mobile phone application and accompanying sound setup.

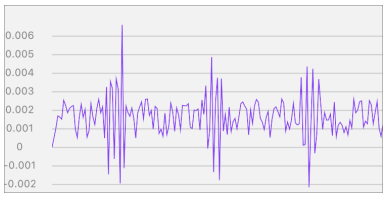
# Chapter 4

# Experiments

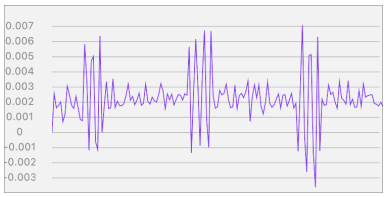
In order to receive the biggest disturbance on the gyroscope readings from our acoustic waves, which we measure by how much movement the gyroscope is registering while lying still, we have conducted a few experiments. In all experiments, the difference in values registered by the gyroscope were most noticeable in the y-axis, so we will mostly be focusing on this axis. When resting on a table with minimal background noise, the y-axis of the gyroscope would stay between the values of around 0.003 and 0.000 radians per second (rad/s). As seen in the graphs further in this chapter, the different tests we ran gave us readings that surpassed these thresholds of 0.003 and 0.000 rad/s. The bigger the deviation from this, the bigger the influence the acoustic waves have on the readings of the gyroscope. The amount of deviation we get in the gyroscope data is important as our application will try to differentiate between moments of disturbance and moments of no disturbance, which will be easier if the deviation is as large as possible, and a large deviation could also ensure we can still notice the disturbance with background noise.

To find out with what setup we can receive the biggest influence on our gyroscopic readings, we conducted experiments on a few variables: the volume, the distance between the phone and the speaker, the angle the sound is coming from, and the frequency of the acoustic waves. For this thesis, we are working with an iPhone 6s. Previous research has already shown us that the resonance frequency for the iPhone 6s is 27.03 kHz [10], thus giving us a starting point.

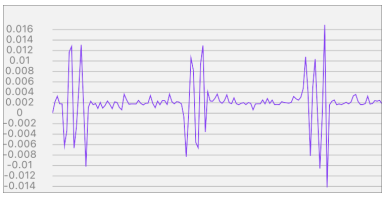
In each of the following experiments, we took one of these variables to test and tried it with different values. Because we only need one axis with noticeable disturbance, we looked at what axis was most influenced and how much radians per seconds we could get. The starting setup is one we already found to be working in earlier tryouts, and we continued each experiment from the best setup found in earlier experiments.




(a) 30 db




(b) 40 db




(c) 50 db

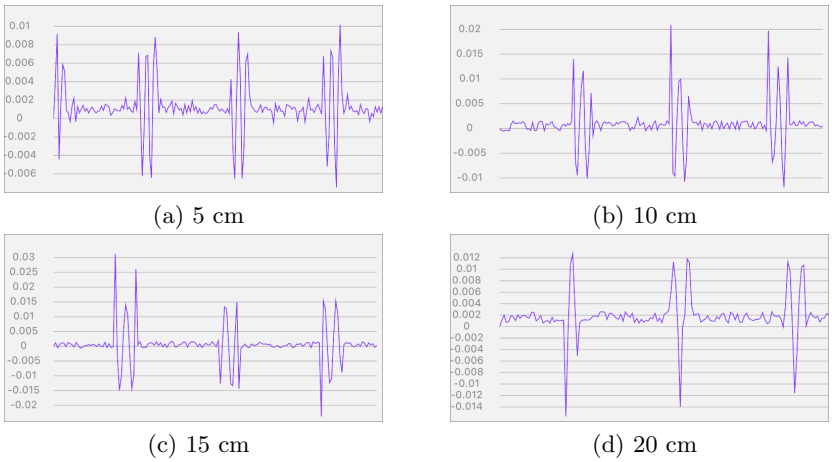
Figure 4.1: Graphs of gyroscope readings being influenced by acoustic waves of different sound pressures

## 4.1 Sound pressure

First, we look at how much the volume of the speaker playing the sounds would influence the gyroscope readings in radians per second. This was a good starting experiment because we already noticed beforehand that sound pressure was a major influence on the magnitude of the distorting waves, thus influencing the readability of the peaks.

The variable in these tests is the sound pressure coming out of the speakers in decibels (dB). While testing, this was done by turning the sound knob, and we used an external decibel meter to turn it into decibels. The different variables are roughly 30, 40, and 50 decibels. This experiment was conducted with the starting values of 27.03 KHz acoustic waves, from a 10 centimeter distance on the right side of the phone.

Figure 4.1 shows us the y-axis of the gyroscope in radians per second. We had the oscillator omit acoustic waves at an interval of two seconds off and one second on. In the graphs, you can see that the higher the sound pressure, the bigger the difference. Acoustic waves with a sound pressure of 50 dB gave us a large enough deviation on the gyroscope readings on the y-axis that the deviation was easily noticed. These tests were done in a lab shared with other people, and we decided 50 dB would be sufficient for the remainder of the experiments, as any louder would cause disturbance to our lab mates (Even though 27.03 KHz shouldn't be heard by a human ear, the speaker still made a humming noise when turned on).




(a) 5 cm

(b) 10 cm

(c) 15 cm

(d) 20 cm

Figure 4.2: Graphs of gyroscope readings being influenced by acoustic waves from different distances

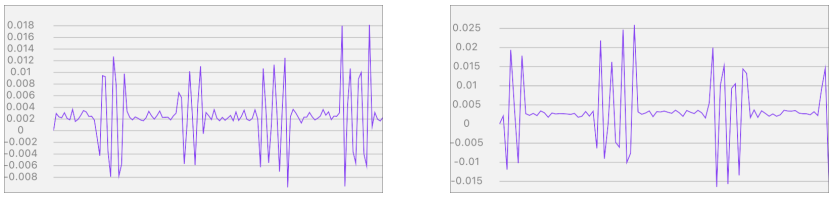
## 4.2 Distance

In our next experiment, we studied the effect of the distance between the speaker that is used to inject acoustic signals and the phone under attack. While the experiment was mainly done to get more information about what would be the best testing setup, it would also give us a lot of information about future possible exploits since not everything can be done from a very short distance. These experiments were conducted with waves of 27.03 KHz from the right side of the phone and a sound pressure of 50dB, as found to be most suitable in the last experiment. The results can be seen in figure 4.2.

The highest response we got was at a distance of 15 cm, where the peaks are measured from 0.028 to -0.02 radians per second. After increasing the distance to 20 cm, the responsibility started to decline again, so we decided that 15 cm was the maximum we were going to work with.

## 4.3 Angle

We also did an experiment to capture how a different angle of where the interfering sound waves are coming from would affect the gyroscope readings. Due to the location of the gyroscope in the phone, this can be an influence. We changed the angle by changing which side of the phone was aimed at the speaker. The results are shown in figure 4.3. In graphs c and d, we show not only the y-axis but all the different axes. We did this because changing the angle also changed the amount of disturbance the different axes had. This can be explained by the fact that gyroscopes sensing mass from different axes respond differently to interference depending on the angle the interference is




(a) Left (b) Right

(c) Top (d) Bottom

Figure 4.3: Graphs of gyroscope readings being influenced by acoustic waves coming from different angles

coming from, since the disturbance is noticed when the sensing mass starts to also vibrate in the axis perpendicular to the axis the sensing mass is vibrating. Even though we now have a sizeable response on the x-axis, the y-axis still has the most deviation, especially when we angle the acoustic waves from the right side.

## 4.4 Frequency

Previous research has already shown us that the resonance frequency of our device, the iPhone 6s, was 27.03 kHz [10]. This was found out by placing the device near a speaker and generating sine waves at frequencies starting from 18 kHz, with increments of 20 Hz. They calculated the average magnitude and thus found out which frequency gave the most magnitude. With this in mind, we still wanted to verify this and test other frequencies close to this frequency to see if maybe we could fine-tune it to get the optimal magnitude possible. The results can be seen in figure 4.4.

As one can see, the small changes in frequency did not give us a noticeable difference. So we decided to stick with 27.03 kHz for the remainder of this thesis.

## 4.5 Conclusion

In conclusion, we get the biggest amplitude with the following setup:

- Volume: 50 dB.
- Distance: 15 cm.


(a) 27.01 KHz

(b) 27.02 KHz

(c) 27.03 KHz

(d) 27.04 KHz

Figure 4.4: Graphs of gyroscope readings being influenced by sound waves of different frequencies

- Angle: From the right.
- Frequency: 27.03 kHz

An overview of this can also be seen in figure 4.5. This is also the setup we will be using throughout the rest of the thesis. This will give us the highest amplitude to work with, making it easier to differentiate between intentional resonance and background noise.

The distance we will be using is too small for a proper tracking application, but with a bigger sound system and a more precise reading algorithm, it could still be a possibility. Note that all these tests were done with a lying-down phone. Tracking a phone being carried by a moving person will cause more complications because this will add a lot of movement to the phone, making it harder for us to notice the intentional interference. But there are certain multiple attack models that can be made using this setup, including a lying-down phone. We will talk about some of these possible applications in chapter 7.

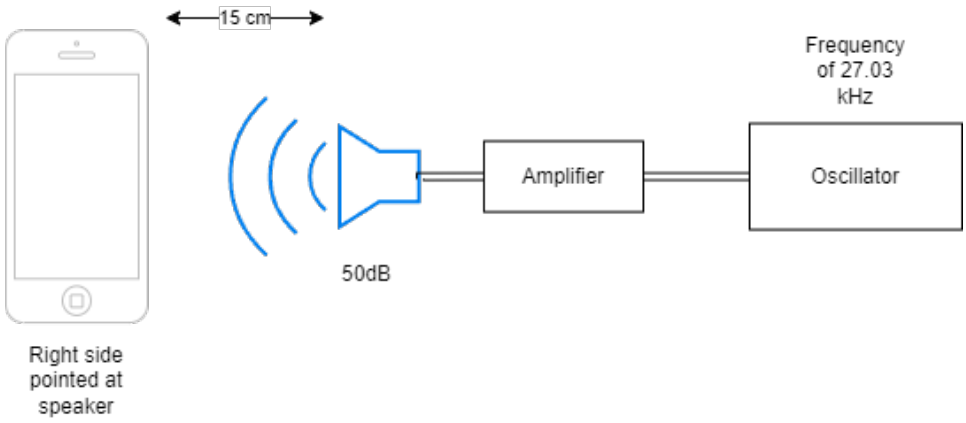



Figure 4.5: Schematic overview of the testing setup with used parameters

# Chapter 5

# Application

In this chapter, we will go over the whole process of developing the stealth communication application, and we will demonstrate how we constructed the acoustic waves. First, we will explain the threat model; after this, we will talk about some important concepts; and in the end, we will discuss how these come together in the implementation of the application. In developing this application, we will use the variables that we collected earlier in chapter 4. In chapter 6 we will run some test cases on the application.

## 5.1 threat model

The application is based on ideas proposed by Nikolay Matyunin et al. [10]. In this paper, they discuss the possibilities of establishing cross-device tracking by using the fact that gyroscope data can be altered by acoustic waves and having an application differentiate between what acoustic waves the gyroscope is picking up. For this thesis, we make this into a real-life implementation. We will create an application that will monitor the gyroscopic data it is receiving, without permission, running on the phone under attack. Separately from this, we create a medium to create acoustic waves in such a way that the gyroscope will pick up these waves once they are in close proximity. When we send these acoustic waves at a variable of intervals, the application will be able to pick up on these different intervals and thus differentiate between the different acoustic waves we generate. We create different intervals for different binary sequences, where a moment of disturbance represents a '1', and a moment of no disturbance represents a '0'. Making us able to turn binary sequences into acoustic waves. The application picks up on these acoustic waves through the phone's gyroscope, and turns the received acoustic waves back into binary. Thus creating a way for us to send binary data to the phone under attack through acoustic waves without needing explicit permission from the phone's user. Different attacks made possible by this threat model will be discussed in chapter 7.

## 5.2 Application

### 5.2.1 Acoustic waves

For creating acoustic waves, we were limited by the equipment that we had. Luckily, the Rigol Function Generator DG822 made it possible to create waves on an interval. Which made it possible to differentiate between acoustic sequences.

Unfortunately, it was not possible to generate a sequence of acoustic waves with changing interval times with this function generator. Which restricted us to only sending repeating sequences of acoustic waves and thus limited us to transferring repeating sequences of binary. While in a real life situation this would probably be too much of a restriction to really get any use out of it, for testing this application it seemed sufficient.

### 5.2.2 Framework

For the framework of the application, we had to find something relatively simple that could also be employed on multiple devices since, in the beginning, it was not totally certain which device would become our main target. With this in mind, we decided to use React-Native[1], as this will give us an application for both iOS and Android operating systems, which covers almost all smartphones. React-Native is written in JavaScript, which also gave us many libraries to use.

### 5.2.3 Data

Our main objective in this application is to get accurate gyroscope data at a small interval. As we mentioned earlier, this data can be received from an Android or iOS device without requiring permission from the user. The library we used for gathering this data was expo-sensors[2]. This library made it possible for us to read the x, y, and z axes of the gyroscope. We store this data inside a state, and once the state refreshes, we append the current value of the state to an array. We have displayed this array on the device itself as a real-time updating graph; this gives us the ability to visually check what is going on and if the data is being received as intended. These graphs can also be used in experiments and test cases.

### 5.2.4 Implementation

Now that we can collect the gyroscope data, we need a way to interpret these acoustic waves and figure out how to transfer them into binary. The idea was to have the audio source send groups of acoustic waves that distort

[1] https://reactnative.dev/

[2] https://docs.expo.dev/versions/latest/sdk/sensors/

the gyroscope readings at different intervals. This way, we could transform periods of distortion into a '1', and periods of no distortion to a '0', leaving us with a binary sequence.

To get this, we needed two things: to differentiate between when the distorting acoustic waves are playing and when they are not, and to time the intervals between new periods of distortion. The first one of these required some testing. We first needed a baseline reading to see between what thresholds the values of the gyroscope data stayed when no distortion was happening. We looked only at the data from the y-axis, as our earlier experiments pointed out that this axis was influenced the most by our acoustic waves. After some trial and error, we figured out that for the iPhone 6s (our testing phone), the y-axis stays between the values of $-0.04$ and $0.08$ when not distorted. So now we know that whenever the value of the y-axis exceeds this threshold, it is receiving our acoustic waves.

The second requirement would start just after we exceed this threshold. We store the beginning and end times of this disturbance, and whenever we get the next disturbance, we can time the interval between two disturbances. Now the only thing left for us to do is map interval sequences to binary. A time of disturbance is mapped to a '1'. And depending on how long the interval between disturbances is, multiple '0's are appended to it, followed by another '1'.

When two disturbances have an interval of 3000 ms, we use that as the standard interval, meaning that the outcome of this sequence would be '11'. Once we have an interval of a length that is twice the standard interval plus the time the distortion is expected, we have encountered the sequence '101'. Lastly, we have three times the standard interval plus the distortion time, which means we have encountered the sequence '1001'.

Because we measured the time between '1's, every sequence we can receive has to start with a '1'. A sequence of only '0's would not trigger our algorithm since no distortion would be noticed. Also, the application now only checks for '1's, 101's, and 1001's, because this gives us more than enough possibilities for this thesis, as it gives the possibility for a lot of different binary sequences. In the future, this can easily be adjusted to allow for even more different sequences.

The application reads the gyroscope continuously, and once it recognizes a sequence in the received gyroscope data, it prints out the binary sequence that is accompanied by this data.

# Chapter 6

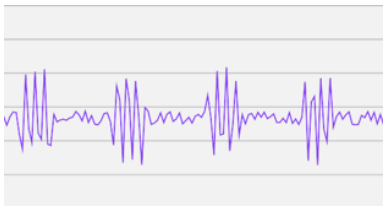
# Test Cases

To make sure the application works as intended, we have conducted multiple tests. A lot of testing was already done while programming the application, but to finalize it, we will do some testing on multiple possible binary sequences and see the probability the application has of recognizing the correct sequence.

The different sequences we decided to do testing on were limited by our equipment as mentioned in the last chapter. The test cases cover all the different combinations of two bits we could come across. The test cases were done with a distance of approximately 15 cm from the right and a frequency of 27.03 kHz. All tests were done on an iPhone 6s. To evaluate a test, we will set up our sound source to emit a sequence of acoustic waves that should be recognized as a certain binary sequence. We will then test to see if the application prints out this intended sequence once the acoustic waves are being played.

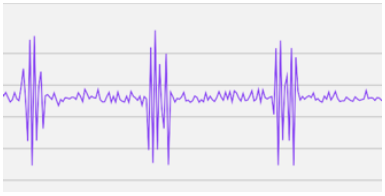
## 6.1 Case 1

For the first test case, we test the sequence '1111'. This would test if a '1' followed by another '1' would be seen as two separate bits instead of one bit. We set up the oscillator so it would transmit one second of waves, followed by two seconds of downtime. After four of these cycles, we should, in theory, have the application recognize an outcome of '1111'. In figure 6.1 A, we can see an example of the waves the application picked up. After receiving these waves, the application maps them to binary and prints out this binary sequence. After five tests, we have a success rate of 100% on recognizing the sequence '1111'.



(a) 1111

(b) 10101



(c) 1001001

Figure 6.1: Graphs from test cases

## 6.2 Case 2

The second test sequence is 10101. This would test if a '1' followed by a '0', and vice versa, would be recognized as such. For this test case, the oscillator was also set up to transmit the acoustic waves for one second, but then it had a downtime of five seconds, in essence skipping one wave. An example of what the gyroscope picked up can be seen in figure 6.1 b. After five tests, we had a success rate of 100% for recognizing the sequence '10101'.

## 6.3 Case 3

For the last test case, we had the sequence '1001001'. This would test if two '0's following each other would be recognized as separate bits. The oscillator setup used for this was one second of waves and eight seconds of downtime, essentially skipping two waves. An example of what the gyroscope picked up can be seen in figure 6.1 C. On the last test, we also had a success rate of 100% after five tests.

## 6.4 Conclusion

We have tested the sequences '1111', '10101', and '1001001', and we had a success rate of 100% for each of these. Due to the equipment, we had to use fairly repetitive test cases, but we did test all different combinations of two bits following each other. Which makes us confident that a less repetitive sequence would yield the same or similar outcomes as these test cases, as the application only looks at small binary sequences. Do note that these tests

were all done on a phone lying flat on a surface, and thus other influences on the gyroscope sensor were limited.

# Chapter 7

# Discussion

In this chapter, we will discuss the outcomes of this thesis. Which consist of: the shortcomings of our application in real-life situations; some actual use cases of the application; we will talk about mitigations that can be taken to secure the MEMS gyroscope, and we will conclude it all.

## 7.1 Shortcomings

The application presented in this thesis has quite a few shortcomings, making it not yet competent in real life. First off, the application was tested and built around a phone that was lying flat on a table. And the reading would be completely off if the phone was in someone's pocket or if someone was walking with it.

Also, there are still only a limited number of possible binary sequences that the application recognizes, although these can be easily expanded when needed. The real shortcoming of this setup relies on the oscillator used. Since it only allowed for set intervals that could not be easily changed while emitting. This results in very few different sequences that can be used with the current setup and application.

The last shortcoming that impacts the usability of our current application is the fact that it was tested and built for a phone that was relatively close to the speaker, approximately 15 cm. We do believe that the application is still functional over longer distances, either with more sound pressure or with a more precise adjusted recognition algorithm.

## 7.2 Possibilities

Even though we can see some shortcomings in our implementation, there are still multiple possibilities for the setup. We could have a laptop communicate with a phone lying next to it through the speakers of the laptop. Which would allow for cross-device internet tracking. Or we could track what

people are watching on television by adding the acoustic waves unnoticed to the audio of different movies. Which can then be picked up by a phone lying on the coffee table. Another possible use case is phones in close proximity, stealthily communicating. And more scenarios involving a target phone and another device capable of emitting sound in close proximity. These scenarios can lead to an unwanted loss of privacy for users.

## 7.3 Mitigation

The obvious mitigation for unnoticed misuse of the gyroscope would be to add a permission layer to it. Since it should be quite obvious which apps need your orientation and which do not, this could prevent a lot of unwanted apps from reading your gyroscopic data. This would, however, still allow apps that often do get permission from users to use the gyroscope, like a video game app that uses movement to steer, to also use the gyroscope for the purposes described in this thesis.

Another mitigation that has been researched by Tao Liu et al.[9], is encapsulating the gyroscope. They showed that putting a case around the gyroscope, made with materials like copper or cardboard, greatly reduces the impact acoustic interference can have on the gyroscope while not limiting the uses the gyroscope is actually needed for. In the same paper, they also propose a scheme that is able to locate potential attackers based on the gyroscope readings.

Sun et al.[14] proposed a filtering algorithm for the MEMS gyroscope that can filter out acoustic interference. The filtering performance can reach up to 95% accuracy, which would also make acoustic interference on the gyroscope way more difficult to pull off than it is today.

And Khazaaleh et al.[8] proposed the idea of monitoring the displacement in the driving direction, and halting the operation of the gyroscope if they detect any large displacement. While this will not prevent from disabling the MEMS gyroscope, it does prevent the attacker from further spoofing the gyroscopic readings.

Since making it secure with software, like adding a permission layer or using filtering algorithms, which can not guarantee full security, and altering the hardware would be a difficult process since the usability of MEMS gyroscopes also depends on their small size, we feel that more research has to be put in to make sure these sensors get more secure in the future.

## 7.4 Conclusion

In this thesis, we showed an application that is capable of receiving acoustic waves through its gyroscope and turning them into binary without permission and without the user knowing that this is happening. Giving us a way to

communicate between two devices while keeping this communication hidden from the user. We can recognize binary sequences with up to 100% success rate. Making it a very reliable option for stealth communication through the phone's gyroscope. While not fully functioning yet in all possible use cases, it does show an important side channel with lots of vulnerabilities. And we brought to light that more protection for sensors like the MEMS gyroscope is definitely needed.

# Chapter 8

# Future Work

For future work, there are many options for improving the attacker model shown in this thesis. With a different speaker setup, the possible communication distance could greatly improve, allowing for much more realistic attack scenarios. Improvements on the algorithm to recognize binary patterns is needed to allow stealth communication to still take place when the target device is moving.

More research into a suitable way to create different binary sequences in the acoustic waves is also needed to really make use of this application. Which would mainly consist of finding or creating the right equipment that makes it possible to program a sequence of acoustic waves with different interval lengths. The application is already capable of receiving these sequences.

We would also like to see more research being done on mitigating the acoustic interference of the gyroscope, or at least big phone producers actually starting to secure these sensors in a way that they need and deserve.

# Bibliography


[1] *ASIA CCS '13: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, New York, NY, USA, 2013. Association for Computing Machinery.

[2] Adam J. Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M. Smith. Practicality of accelerometer side channels on smartphones. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, page 41–50, New York, NY, USA, 2012. Association for Computing Machinery.

[3] Liang Cai and Hao Chen. TouchLogger: Inferring keystrokes on touch screen from smartphone motion. In *6th USENIX Workshop on Hot Topics in Security (HotSec 11)*, San Francisco, CA, August 2011. USENIX Association.

[4] R. N. Dean, G. T. Flowers, A. S. Hodel, G. Roth, S. Castro, R. Zhou, A. Moreira, A. Ahmed, R. Rifki, B. E. Grantham, D. Bittle, and J. Brunsch. On the degradation of mems gyroscope performance in the presence of high power acoustic noise. In *2007 IEEE International Symposium on Industrial Electronics*, pages 1435–1440, 2007.

[5] Petter Hallmo, Arne Sundby, and Iain W. S. Mair. Extended high-frequency audiometry: Air- and bone-conduction thresholds, age and gender variations. *Scandinavian Audiology*, 23(3):165–170, 1994. PMID: 7997833.

[6] Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, and Joy Zhang. Accomplice: Location inference using accelerometers on smartphones. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pages 1–9, 2012.

[7] Zhen Hong, Xiong Li, Zhenyu Wen, Leiqiang Zhou, Huan Chen, and Jie Su. Esp spoofing: Covert acoustic attack on mems gyroscopes in vehicles. *IEEE Transactions on Information Forensics and Security*, 17:3734–3747, 2022.

[8] Shadi Khazaaleh, Georgios Korres, Mohammed Eid, Mahmoud Rasras, and Mohammed F. Daqaq. Vulnerability of mems gyroscopes to targeted acoustic attacks. *IEEE Access*, 7:89534–89543, 2019.

[9] Tao Liu, Zhen Hong, and Huan Chen. A traceability localization method of acoustic attack source for mems gyroscope. *IEEE Embedded Systems Letters*, 15(1):13–16, 2023.

[10] Nikolay Matyunin, Jakub Szefer, and Stefan Katzenbeisser. Zero-permission acoustic cross-device tracking. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 25–32, 2018.

[11] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1053–1067, San Diego, CA, August 2014. USENIX Association.

[12] 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 amp; Applications*, HotMobile '12, New York, NY, USA, 2012. Association for Computing Machinery.

[13] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 881–896, Washington, D.C., August 2015. USENIX Association.

[14] Yufei Sun, Peng Guo, Lihui Feng, Chaoyang Xing, and Junjie Wu. A filtering algorithm of mems gyroscope to resist acoustic interference. *Sensors*, 20(24), 2020.

[15] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 3–18, 2017.

[16] Jeff Watson. Mems gyroscope provides precision inertial sensing in harsh, high temperature environments.

[17] Junjie Wu, Yufei Sun, Peng Guo, Lihui Feng, Yongbin Zhang, and Youqi Zhang. Resonance interference research of mems inertial sensors and algorithm elimination. *IEEE Sensors Journal*, 22(11):10428–10436, 2022.

# Appendix A

# Accessing the code

This thesis is centered around the development of an iOS application. To access this application, go to `https://github.com/JelleMedendorp/TrackingApp`, and clone the repository.

To run the application, we made use of Expo, which can be downloaded from `expo.dev`. Once Expo is installed, go to the map where the repository is cloned and run:

```
expo install
expo start
```

Which will run the application in the browser. To test it out on a phone, one can install the Expo app on the phone in the App store(iOS) or play store(Android). To run the application on this device, scan the qr code seen in the cmd shell with the phone's camera.