An NCC Group Publication

An Introduction to Ultrasound Security Research

Prepared by:
Alex Smye

Contents

1 Introduction

Over the past few years there has been an increase in the use of sound as a communications channel for device-to-device communications. This practice has been termed Data-Over-Sound (DOS) and has been billed as a cheap and easy to use alternative to traditional communications protocols such as Wi-Fi and Bluetooth. As this is a relatively new technology, it is lacking any kind of standardisation, while different manufacturers and companies offer competing technologies with different implementations and varying levels of security. This means that as this technology gains more momentum there are likely to be a number of security questions raised and assurances sought.

Of particular interest is the use of Ultrasound (20kHz+) or Near-Ultrasound (16kHz-20kHz) for the purpose of device to device communications. These frequencies are outside the hearing range of most adults and so ultrasound or near ultrasound can be used as a covert communications channel. Manufacturers and developers favour these frequencies due to improved noise resilience and bandwidth in comparison to audible frequencies. As a result, the majority of DOS technologies and use cases use these frequency ranges. This paper examines the use of Ultrasound and Near Ultrasound as a communications channel and evaluates potential security issues within them.

2 Background

Our research initially looked at the use of ultrasound across the entire ultrasound spectrum. However, the use of true ultrasound (24kHz+) was mostly restricted to imaging and sensing use cases. As these frequencies do not have many uses in a security context, there was little reward identified from exploitation. Additionally, exploring and exploiting these frequencies would require specialized hardware. All of the use cases identified in this whitepaper used near-ultrasound in the range of 18 kHz-24 kHz, and as such utilise existing, non-specialised hardware, namely speakers and microphones.

This frequency band was found to have many more existing uses for potentially sensitive operations such as payments and device onboarding. As such, the near ultrasound band was the focus of this research. Near ultrasound and ultrasound will be used interchangeably in this whitepaper and both are referring to the near-ultrasound range unless otherwise specified.

The use of near ultrasound is surprisingly widespread and has a number of advantages over traditional communication protocols (Bluetooth, NFC, Wi-Fi etc.). The main attraction of ultrasound as a communications protocols is the compatibility of hardware. The majority of mobile phones, laptops etc. are manufactured with integrated microphone and speakers as standard. The only restriction on the use of a specific speaker and microphone pair is their different frequency responses. Essentially, they must both be able to emit and receive the high frequency sounds that are required for this kind of communication. This restriction is more prevalent in microphones due the large variety in the quality and size of the devices that are available in the market. However as the diagram below shows, the frequency response of the microphone of modern mobile handsets allows the device to listen in to frequencies up to around 14kHz without significant loss. Furthermore, although the behaviour at frequencies above 14kHz is somewhat undefined, the frequency response is still sufficient for the handset to listen to frequencies up to about 24kHz.

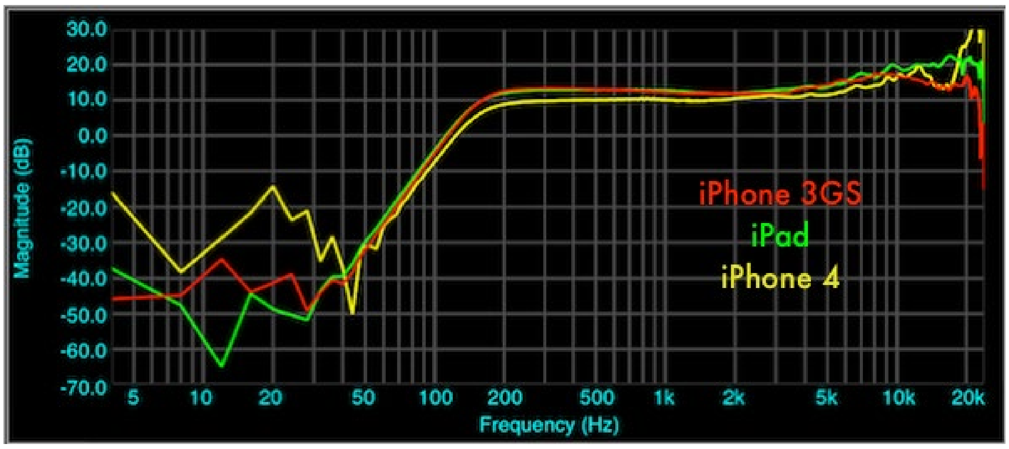


Figure 1 - Frequency Response of Mobile Handset Microphones [1]

Interestingly, using the manufacturer supplied headset rather than the integral microphone gives improved frequency response at the higher end of the near ultrasound range. This is likely the case across the board as headsets often contain better quality microphones and audio hardware. The use of headsets for ultrasound communication is further explored in *MOSQUITO - Covert Ultrasonic Transmissions between Two Air-Gapped Computers using Speaker-to-Speaker Communication* [2].

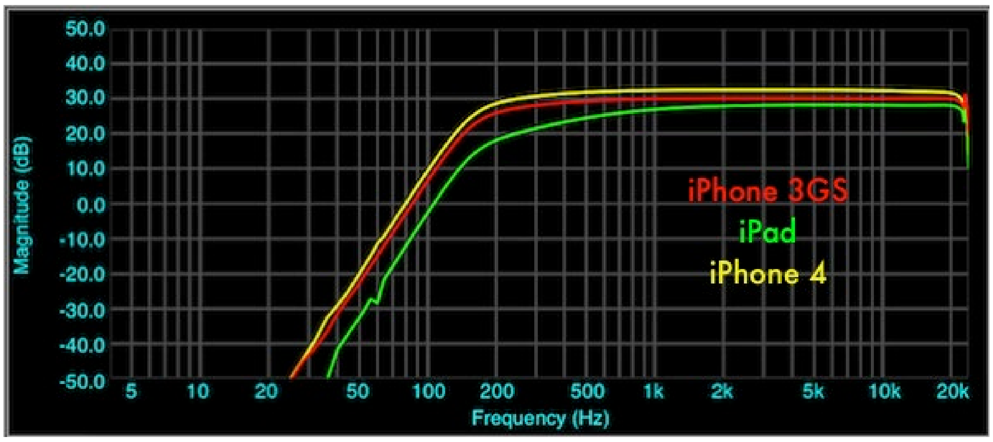


Figure 2 - Frequency Response of Headset Microphone [1]

As explained previously, in the majority of cases, a near ultrasound communications channel utilises a device's existing hardware. As such, there is no additional cost associated with using near ultrasound for communications. Whilst a large number of consumer devices are now manufactured with a large array of communications hardware (Wi-Fi, Bluetooth, NFC etc.), IoT devices and other low cost hardware may not have this specialised hardware installed as default in an effort to keep costs low. As a result, a speaker and microphone pair may provide a cheap alternative.

For the majority of speakers, they will have no trouble emitting sounds in the near ultrasound range. This is a design requirement in order to accurately play high quality sound. However, the extreme case in this scenario is the use of mobile device speakers for near ultrasound emission. These speakers are often small and there is less of a requirement for high quality audio. However, the diagram below, showing the audio output from an iPhone 7 speaker, shows that even in this extreme case, although not optimised, near ultrasound emission is still possible.

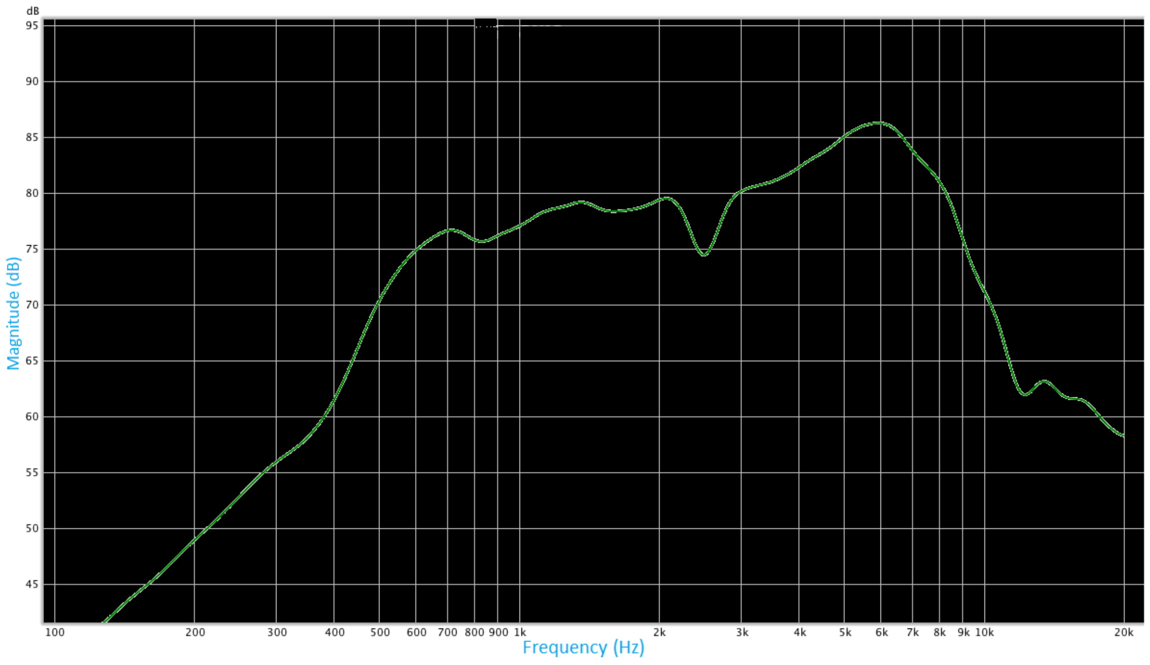


Figure 3 - Frequency Response of Mobile Handset Audio Output

The diagrams above demonstrate that the emission and reception of near ultrasound is possible for the majority of electronic devices even though this falls outside the requirements for many devices. As a result, near ultrasound is a viable channel for device-to-device communications.

The use of near ultrasound as a communications mechanism uses simple existing hardware. In addition to this, the physical layer of the communications channel is relatively simple; all a developer needs is access to a device's microphone and speaker. Building a simple communications channel on top of this is straightforward.

The use cases below were identified from research into existing and proposed technologies and products. It is in no way exhaustive and brand new use cases appear regularly:

- Proximity verification
- Device onboarding
- P2P payments
- Offline communications
- Authentication
- Data exfiltration from air gapped machines

From these use cases, three generalised categories of use case were envisaged; **user tracking**, **data communications** and **data exfiltration (Covert Communications)** and these are further explored in this whitepaper.

3 User Tracking

3.1 Overview

Most identified use cases of user tracking using ultrasound involve the use of a beacon and receiver. The beacon emits an ultrasound frame containing beacon-specific data. This beacon is picked up by a receiver (in most use cases this is a mobile phone), which processes the ultrasound frame and extracts the data. These beacons can be distributed using any method that utilises hardware with the ability to play high frequency audio. In this way, a user's proximity to a beacon and therefore an approximate location can be inferred. This has an obvious application in device onboarding and authentication through proximity. However, this technology is also used to track users and can be used to determine user behaviours such as a user visiting a certain shop or watching a certain TV show.

Once the ultrasound frame has been received, the ultrasound-enabled receiver performs actions based on the data that is received. It is up to the ultrasound framework or application installed on the receiver to dictate what actions are performed upon receipt of an ultrasound beacon. The majority of tracking frameworks send analytics and data back to the tracking provider upon receipt of a beacon, often including information such as the user's location, email address, android device ID and other potentially sensitive information. With this information, the tracking provider can then serve the user tailored advertisements or simply collect large amounts of user data to build a user profile. Examples of this kind of tracking include:

- Physical inaudible beacon emitter in locations of interest
- Other ultrasound-enabled devices (Laptops, Google Home etc.) emitting inaudible beacons
- TV advertisements containing inaudible beacons

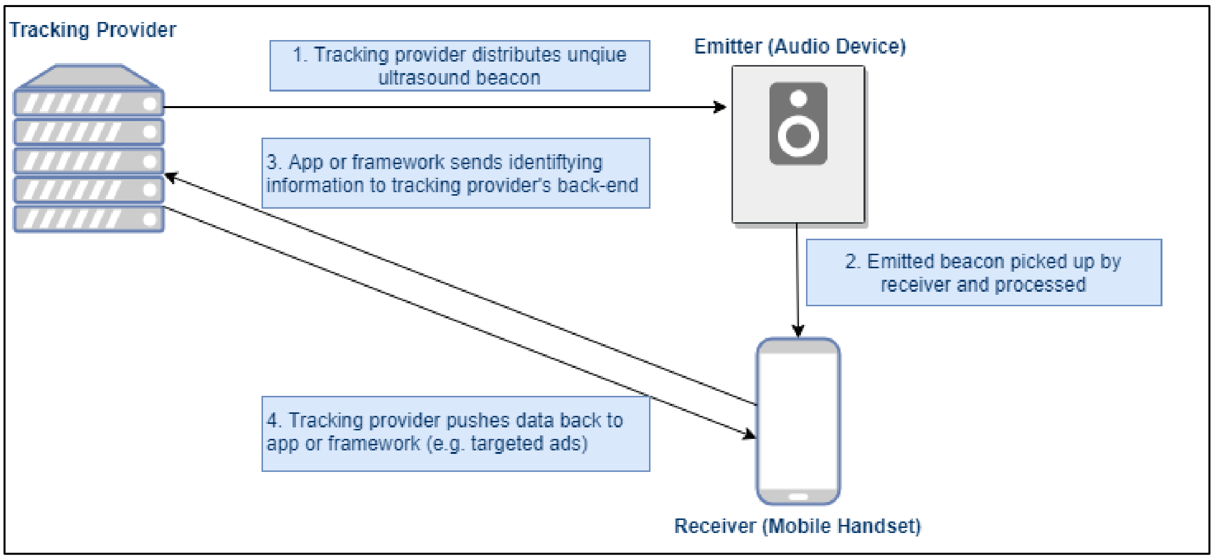

Tracking Provider
1. Tracking provider distributes unqiue ultrasound beacon
Emitter (Audio Device)
3. App or framework sends identiftying information to tracking provider's back-end
2. Emitted beacon picked up by receiver and processed
4. Tracking provider pushes data back to app or framework (e.g. targeted ads)
Receiver (Mobile Handset)

Figure 4 - Ultrasound Device Tracking (UDT)

The receiving ultrasound framework may perform a number of different actions upon receipt of an ultrasound beacon. The behaviour of each framework and use case is different; however below are some general examples of identified behaviours:

- Receiver sends back analytics and user info to the tracking provider including location in order to build a user profile
- Receiver is sent tailored ads based on the location and user profile (URLs may be included in beacons directly, for the receiver to fetch – an obvious security risk)

A number of companies go further than simple tracking and offer ultrasound cross device tracking solutions (uXDT). These solutions can track a user between their devices. A diagram demonstrating how this works is shown below:

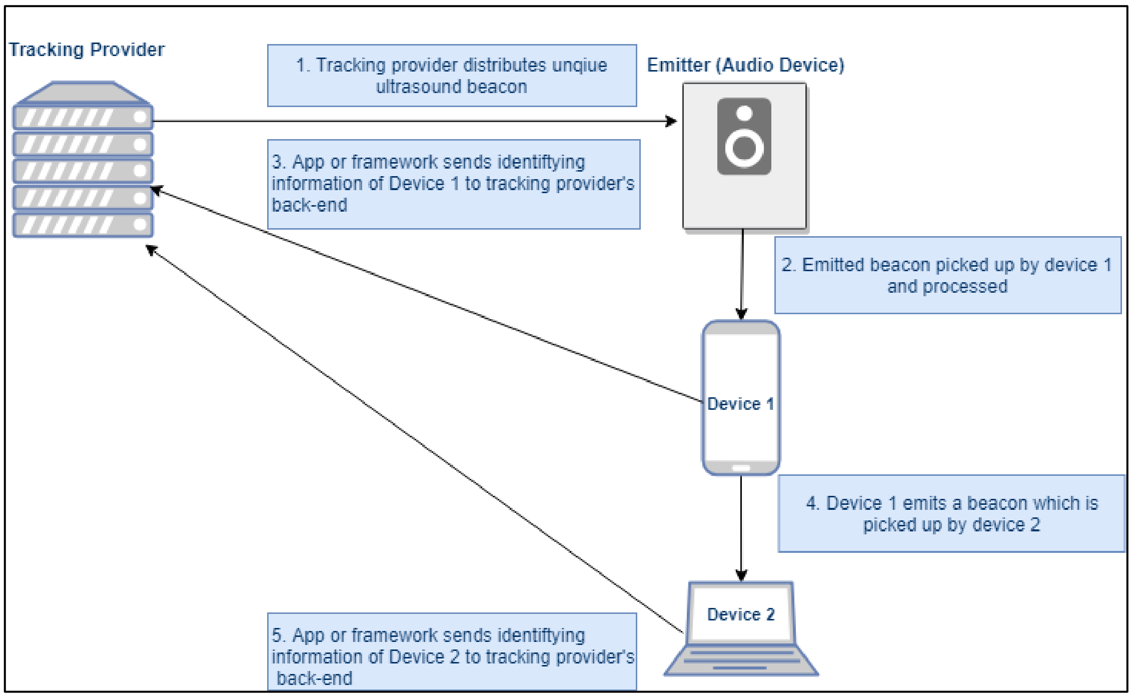

Tracking Provider
1. Tracking provider distributes unqiue ultrasound beacon
Emitter (Audio Device)
3. App or framework sends identiftying information of Device 1 to tracking provider's back-end
2. Emitted beacon picked up by device 1 and processed
Device 1
4. Device 1 emits a beacon which is picked up by device 2
Device 2
5. App or framework sends identiftying information of Device 2 to tracking provider's back-end

Figure 5 – Ultrasound Cross-Device Tracking (uXDT)

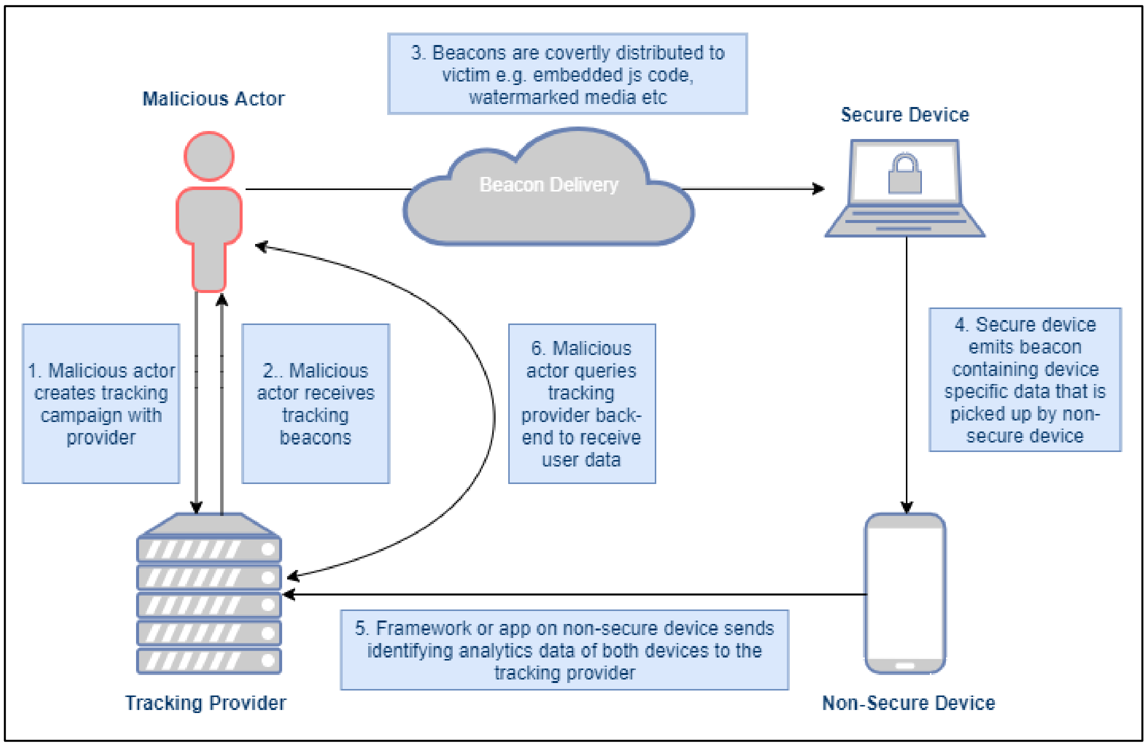
An advertising client will go to a cross device tracking service provider, which will generate ultrasound beacons that will be used for this purpose. These beacons are then distributed using any method that the advertiser sees fit. The user's device, in this case their phone, picks up this beacon and reacts in the usual way, contacting the uXDT provider. However, this device is also used as a beacon itself to be picked up by other ultrasound-enabled devices in the vicinity. This exchange of data between the two devices effectively links the user to numerous devices and helps build a more detailed user profile.

Two of the most popular ultrasound tracking frameworks are Shopkick and Silverpush. It should be noted that in 2016 Silverpush received a warning from the FTC in the United States [3] stating privacy concerns. Consequently, Silverpush removed the ultrasound tracking functionality from their software in the United States, however it is not clear if this removal applied to other locations. In addition to this, the FTC warned app developers whose apps contained Silverpush code of potential privacy concerns within their app. Google also removed apps from the Google Play store that it found contained Silverpush code. However, recent research [4] conducted in 2017 found 234 apps still contained Silverpush code. The current use of these frameworks within application is unknown however given the previous trend of increased usage; it is likely to have increased.

3.2 Weaknesses

On the Privacy and Security of the Ultrasound Ecosystem [5] lays out four major areas of weakness and exploitation for the user tracking use case:

De-Anonymization attacks – In a similar methodology to the uXDT, an anonymous user is de-anonymized using their anonymous device as a beacon. As before, any method that is able to use the device's speakers to emit ultrasound is viable. For example, an anonymous user may visit a website with some JavaScript code embedded which plays a sound, emitting an ultrasound beacon from the device's speakers. Alternatively an anonymous user listens to media with hidden inaudible beacons embedded in the audio. A nearby device, which is not secured to the same degree as a hardened computer/Tor-enabled browser, with some form of ultrasound-enabled framework or application can listen for and pick up this beacon. Once this beacon has been processed on the non-secure device it will communicate with the tracking provider's back-end with details of the secured device and effectively link the two devices, de-anonymizing the user. An example case is shown below, whereby an adversary obtains a valid uDXT token from a tracking provider and uses the existing cross-device tracking ecosystem to de-anonymize the user.



3. Beacons are covertly distributed to victim e.g. embedded js code, watermarked media etc
Malicious Actor
Secure Device
Beacon Delivery
1. Malicious actor creates tracking campaign with provider
2.. Malicious actor receives tracking beacons
6. Malicious actor queries tracking provider back-end to receive user data
4. Secure device emits beacon containing device specific data that is picked up by non-secure device
5. Framework or app on non-secure device sends identifying analytics data of both devices to the tracking provider
Tracking Provider
Non-Secure Device

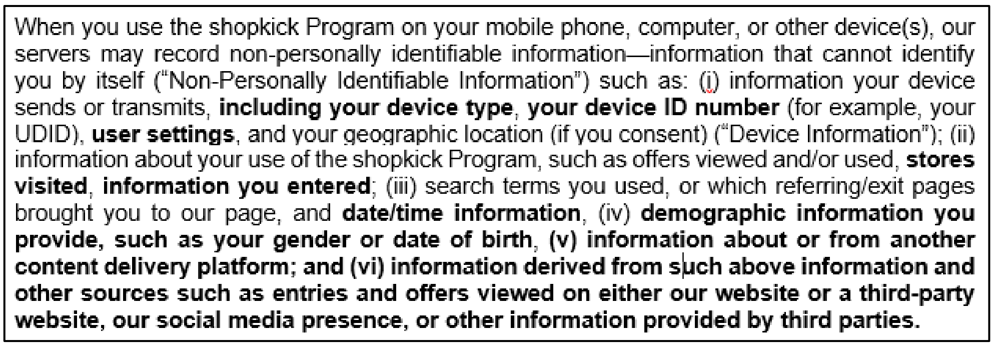
Figure 6 - De-Anonymization Attacks

It is worth noting that it would be possible to remove the tracking provider completely if a malicious actor could get their own app/code on the non-secure device. This malicious app or code could communicate directly with the malicious actor.

Beacon injection – The way in which a particular framework or application handles data is specific to that particular code and the majority of these frameworks and apps do not release this information or code. As such, if there are vulnerabilities in the code which processes the data contained within the ultrasound beacon, it may be possible to inject data into beacons which could cause undefined behaviour and lead to exploitation. Theorised examples of this are shown below:

- A beacon contains a URL of an ad for an app to fetch and display to the user. If it was possible to create a valid beacon and enter an arbitrary URL, it may be possible to direct unsuspecting users to a malicious URL.
- A beacon contains commands for an app to execute some kind of command such as ping the tracking provider's backend. If it was possible to create a valid beacon containing malicious code it may be possible to inject commands or code into the application or even the back-end services to which the app talks.

Information leakage – The ultrasound enabled application that act as the receivers in this instance often send back user and analytics data to the tracking service provider. An extract from the privacy policy of a popular tracking framework, Shopkick, is shown below. This extract is in relation to the "Non-identifiable information" that Shopkick is allowed to collect;



When you use the shopkick Program on your mobile phone, computer, or other device(s), our servers may record non-personally identifiable information—information that cannot identify you by itself ("Non-Personally Identifiable Information") such as: (i) information your device sends or transmits, **including your device type**, **your device ID number** (for example, your UDID), **user settings**, and your geographic location (if you consent) ("Device Information"); (ii) information about your use of the shopkick Program, such as offers viewed and/or used, **stores visited**, **information you entered**; (iii) search terms you used, or which referring/exit pages brought you to our page, and **date/time information**, (iv) **demographic information you provide, such as your gender or date of birth**, **(v) information about or from another content delivery platform; and (vi) information derived from such above information and other sources such as entries and offers viewed on either our website or a third-party website, our social media presence, or other information provided by third parties.**

Figure 7 - Extract from Shopkick's Privacy Policy

As can be seen above, the framework collects a large amount of sensitive information about users. Perhaps most concerning are parts **V)** and **VI)**. These parts reveal that Shopkick collates user information from other platforms and sources in order to make an even more detailed user profile. With any amount of sensitive information, there is always a risk of leakage of this information either in transit or at rest. There have been confirmed cases of this information being leaked in transit within the Silverpush framework. Sensitive information was sent to the back-end services using unencrypted HTTP connections which can be trivially sniffed [5], [6], [7].

Beacon replay – Due to a lack of authentication within certain beacon's protocols, it is possible to capture beacons and replay them whenever and wherever you like. This could be exploited if a particular beacon is associated with a malicious ad campaign. A malicious user captures this beacon, then goes into a say a coffee shop, and replays this. All enabled receivers will acknowledge this beacon reception with the tracking provider's backend will be served target ads. These ads will likely have some revenue associated with them and hence the malicious user gets revenue from the malicious ad campaign. Additionally, this hosted ad campaign could push malicious ads to the user.

4 Data Communications

4.1 Overview

Due to the low bandwidth and noise susceptibility, traditionally, ultrasound has not been used as a traditional communications channel. However, ultrasound is becoming increasingly used as a cheap and easy to use alternative to traditional communications protocols such and Wi-Fi or Bluetooth.

There are a number of limitations on the use of ultrasound as a communications channel. Firstly the range of communication is a serious limiting factor. Due to the nature of sound waves they do not travel well over large distances. Attenuation reduces the signal power dramatically in air over larger distances. Additionally sound waves cannot penetrate surfaces in the same way that radio waves can, further reducing the range of ultrasound communication. Secondly, due to the low carrier frequency (~21kHz), upper limits of the hardware (~24kHz) and the lower limit due to the audible frequency range (~18kHz) the usable bandwidth is relatively low. This reduces the capacity of the channel. Another challenge is noise. In modern day environments, there is a large amount of noise from a variety of sources. For example, electronic devices often emit sounds in the ultrasound and near ultrasound range. These noise sources interfere with any ultrasound based communications channels in the vicinity causing data to be lost, rendering the communication useless. The final limitation is the sampling rate of the hardware involved. Sampling theorem dictates that in order to accurately represent an analogue signal from a digital source the sample rate must be at least twice the highest frequency in the signal to be reproduced. As the sample rate of electronic devices is usually fixed at 44-48kHz, the upper frequency limit for both the recording and emission of near ultrasound is 22-24kHz. Certain devices may not have a high enough sample rate in order to accurately reproduce these signals.

Despite the limitations of ultrasound as a communications channel, there are a number of existing frameworks and applications that have been developed for this purpose. The four of the most popular are Chirip.io, Copsonic, Lisnr and Google Nearby. Each has their own proprietary communications protocol and further research is needed to look into these protocols in order to understand how they work and if any vulnerabilities exist. This is further complicated by the fact that, thus far, these solutions have employed a security through obscurity approach and detailed technical details are sparse. However, it is possible to derive details from technical documentation and other sources. A technical summary of different data communications over ultrasound technologies is shown below:

Framework	Communications Protocol	Frequency Range	Data Rate	Security / Authentication
Chirp	FSK	18.25-20kHz	Up to 1kbps	No authentication or security by default
Copsonic	Proprietary Protocol – "Not DTMF, nor PSK, nor FSK"	Unknown - Likely 18-20kHz	Up to 15kbps (When using full audio spectrum)	Unknown - Proprietary Protocol
Lisnr	FSK	18.75-19.2 kHz	Up to 1kbps	Optional encryption
Google Nearby Messages	FSK with DSSS	18.5–20.5khz	Unknown	No Authentication or security

After investigation of these existing near ultrasound communications frameworks and applications, the following general characteristics were identified (Note that this is not necessarily true for all frameworks and applications):

- **18 kHz– 21kHz** – This frequency range is above the audible frequency range and below the upper limit imposed by hardware sample rates, giving an effective bandwidth of 4kHz.
- **Frequency Shift Keying (FSK) modulation** – This simple modulation is chosen due to its resistance to noise. In general, other modulation schemes rely on changes in amplitude or phase of a carrier and ultrasound communications channels are not well suited to ultrasound communications channels due to their noise and other attenuating factors. A number of distinct frequencies within a range are chosen, with a given frequency representing a symbol. Each symbol corresponds to a selection of bits. In this way, data is sent over the air from an emitter to a receiver. A visual representation of this is shown below. This frame was captured when data was sent using the Chirp.io framework. As can be seen, there are distinct frequencies that are transmitted at a set interval or period.

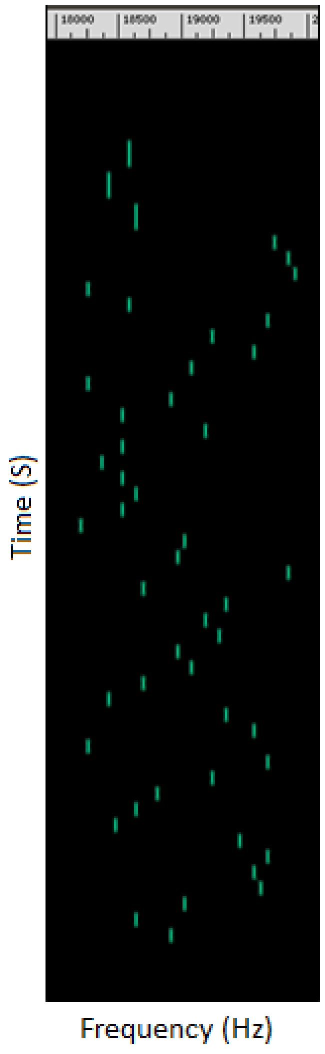


Figure 8 - Captured Chirp Frame

- **Modest bitrate** – The majority of solutions and products that were investigated had low effective bitrates. Whilst some vendors advertise bitrates of 1kbps or higher, these are only achievable in specialized conditions or when using higher bandwidths, stretching into the audible frequency range. In reality, once adequate error correction and other required redundancy is added, the bitrate is much lower. For example, Chirp's ultrasonic framework has an effective bitrate of around 16bps.

- **Error Resilience** – Due to the high levels of attenuation and vulnerability to noise, robust error correction is used to reduce the effect of errors introduced in transmission. Simple error detection would not suffice in this instance as duplex communication is not common and so the receiver cannot trigger a retransmission. Forward error correction adds redundancy to the transmitted data frame, enabling the receiver to 'reconstruct' frames, which contain errors.

4.2 Weaknesses

There are a number of weaknesses associated with the use of ultrasound as a communications channel.

Due to the very localised nature of ultrasound and the simplicity of hardware, it would be easy to create a localised jammer. This would only require a simple speaker or audio source to effectively block all ultrasound communications within a room or small area. As the ultrasound channel is highly susceptible to noise, a jammer of this nature would likely cause widespread denial of service conditions.

As is also the case with IoT there is an element of manufacturer complacency when considering the security of ultrasound communication products. Manufacturers of ultrasound communications technologies seem to rely on the localised nature of ultrasound to offer a layer of security. An attacker needs to be in close proximity to devices in order interact with them or pick up the ultrasound emissions. In addition to this, manufacturers are currently relying on a security through obscurity approach, very few companies release detailed information about how their products work, and the security measures in place, on the assumption that their technologies cannot be reverse engineered. Both these assumptions are dangerous and, with enough time and effort, an attacker can reverse engineer the technology and overcome range issues, for example, by using relay attack. Indeed, there are examples of attempts to reverse engineer Google Nearby [9] and Chirp.io [10] in the public domain.

Due to the focus on low cost and very short time to market of ultrasound solutions, it seems that security is often added as an afterthought, or not at all. In addition to this, the low bandwidth of the channel means that overheads for authentication and encryption may be considered too high and these important security features may be left out of the finished product. This puts devices at risk of Man-in-the-Middle (MitM) attacks, replay attacks and sniffing of sensitive information. It should be noted that some companies (CopSonic, Lisnr, etc.) offer "Secure" solutions for P2P payments, remote authentication and data transfer. However, as explained above, these projects are closed source, and details are sparse. Further work needs to be done to evaluate how secure these solutions actually are.

5 Data Exfiltration (Covert Communications)

There have been numerous cases of acoustic side channels being used for data exfiltration in novel research. This research leverages the fact that nowadays, although secure devices are likely to have network and external connectivity removed, a large number of modern devices have integrated speakers/microphones or peripheral devices capable of making acoustic sounds (HDD [11], fans [12] etc.). The majority of these papers envisage a scenario where an attacker is attempting to exfiltrate data from a secure, air-gapped computer using acoustic side channels. Whilst the focus of this research has been ultrasound, other acoustic channels will act in the same manner and face the same challenges.

There are a number of limitations to using ultrasound as a side channel for data exfiltration, firstly, as has been explained previously, ultrasound is very localised and doesn't travel well over longer distances or through objects such as walls. As a result, an attacker would have to be in close proximity to the target. If the target is physically isolated this may not be possible. Secondly, the channel capacity of an ultrasound communications channel is small. As stated earlier, the average bitrate that was found when evaluating existing products and research was around 50bps. Whilst this is not a problem for a small amount of data (A 2048-bit RSA key could be extracted in around 40 seconds), it is unlikely that large files could be extracted reliably. An additional complication with large files is the likelihood of errors in transmission. As with all cases of attacking secure, isolated devices, the biggest problem is likely to be the initial infection or initial access to the device. A secure device is likely to be segregated from any networks and stored securely, away from public view, making it hard to access. This isolation also means that communication with any kind of Command and Control (C&C) infrastructure would be extremely limited. This would make the management and exfiltration aspect of any successful infection difficult to manage. However, a dedicated attacker could overcome this problem given enough time and effort, an example of this is Stuxnet, which infected air gapped machines through infected USB sticks. The final issue is that the target devices may not have the required hardware, a speaker or microphone, attached. This could limit exfiltration to one-way communication or restrict it completely.

There have been a number of papers that have sought to overcome these limitations with novel techniques, the two most comprehensive are *MOSQUITO: Covert Ultrasonic Transmissions between Two Air-Gapped Computers using Speaker-to-Speaker Communication* [2] and *On Covert Acoustical Mesh Networks in Air* [13]. In these papers, both develop bespoke communications protocols to overcome the limitations above. *On Covert Acoustical Mesh Networks in Air* adapts an underwater communications protocol, GUWAL (Generic Underwater Application Language), to attempt to improve the error resilience, and therefore the range of communications. This paper envisions a mesh network of infected devices; much like UxDT discussed in Section 3.1, and employs the use of GUWMANET (Gossiping in Underwater Mobile Adhoc NETworks) at the network layer to enable mesh networking. By using a mesh network the exfiltration of data the range, speed and resilience of the extraction can be maximised. Additionally this paper envisions an "End device" which is part of the mesh network, and is connected to the internet, allowing for external C&C.

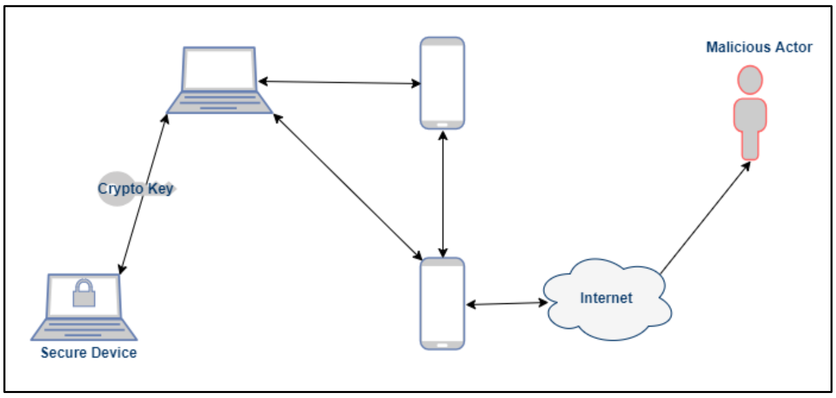


Figure 9 - Diagram of Acoustic Mesh Network

MOSQUITO: Covert Ultrasonic Transmissions between Two Air-Gapped Computers using Speaker-to-Speaker Communication also outlines the development of a bespoke communications protocol, but it goes further and details the requirements of malware that would be used for this purpose. In particular, they discuss how malware could overcome limitations of hardware by, for example, repurposing audio jacks and using an inefficient yet viable speaker to speaker communications channel.

The majority of work in this areas of research is heavily theoretical or requires specialised conditions to be considered a viable means of data exfiltration. The time and effort required to create viable ultrasound data exfiltration methodology means that this is only a real concern to nation states or very high value targets. The specialised requirements of proximity, initial access and autonomous operation mean that it is simply not worth the effort for attacker to target anything other than the most valuable of targets using this method. Besides the contentious, alleged BadBIOS malware discovered by Dragos Ruiu [14], there have been no occurrences of data exfiltration using ultrasound revealed in the public domain but it is difficult to know how widespread this practice is as any potential attackers or targets are likely to be extremely secretive. However, in this whitepaper, ultrasound has been shown as an effective solution for data communication and exfiltration. Therefore, these high value targets should consider this a real risk and have suitable mitigations in place.

6 Conclusion

With the growth of IoT, interoperable devices and the current focus on low-cost easy-integration hardware, the use of ultrasound as a communications channel is likely to increase. The rate of increase is dependent upon a number of factors. *Disruptive Analysis'* analysis [15] of generic Data Over Sound (DOS) technologies identifies the following key requirements for mass uptake of DOS as a viable communications channel. Firstly, the adoption relies upon standardisation. Currently DOS are vendor specific and vary greatly between framework and applications, therefore different frameworks and application are not interoperable. An interoperable standard would likely increase the adoption and make sure that technologies adhere to specific security and ethical guidelines. Secondly, technologies should be made open source in order to drive innovation and adoption. Innovation and increased adoptions will, in turn, increase training and skills within the area. Thirdly, in order to see widespread use, DOS needs to be an embedded feature in various platforms, whether this be IoT frameworks or device OS's. Making this technology easily accessible and usable will help encourage developers and manufacturers to consider DOS as a useful tool.

Whilst we are still a way of widespread adoption, there are a variety of use cases that have already emerged for DOS as a communications channel. Some of these are relatively innocuous (i.e. Ultrasound for communication in toys such as Furby) and there is little concern about security in these cases. However, there has been a growth in the use of ultrasound for more sensitive use cases, such as P2P payments or device on-boarding. As the use of DOS increases, the security concerns and requirements will become more important and will have to be realised. However, unfortunately, due to the lack of standardisation and competing technologies, each company's technology will have to be investigated separately.

There are a number of preventative measures that could be used to reduce the risk to users and devices from ultrasound exploitation, through blocking of the communications channel entirely and standardisation.

- **Filtering** – As in traditional Digital Signal Processing (DSP), it is possible to filter out certain frequencies of sound in hardware or software. Software would be the easiest method as it would not require and hardware changes to a device. A simple low pass filter, cutting out frequencies above 16kHz would be relatively easy to implement in software, however it is likely that a rooted or jailbroken device would be needed for mobile devices. uBecSec.org have developed android and chrome extensions for this purpose, however this only applies for web browsing and the filtering is not applied at the system level
- **Permissions** – Most mobile application have a set list of permissions they require in order to function. In terms of microphone permissions, both iOS and android have blanket microphone permissions which effectively allow an application access to the entire audio spectrum. As a result, any app that requests access to the microphones can listen for near ultrasound frequencies. An effective mitigation would be to develop an ultrasound or inaudible communication permission that would require a user to explicitly allow an application to use the microphone to listen to frequencies above 16kHz. However, the method by which access to frequencies above 16kHz would be blocked is not immediately clear. This could be achieved using the filtering methods as described above.
- **Jammers** – A wideband jammer would be easy to implement, all that would be needed is a speaker and some kind of signal generator emitting noise in the 16-24kHz range. This would effectively block all communication/tracking by drowning out the data signal. There are health concerns with this though; near ultrasound frequencies can have negative impacts on a person's health.
- **Scanners/Detectors** – Scanners or detectors of near ultrasound frequencies are relatively easy to create and use. The issue with this is that this method only detects the presence of ultrasound, it doesn't stop communication or tracking. A python script that can be used to detect near ultrasound and general purpose receiver can be found on the wiki.
- **Standardisation** – Currently all the applications and frameworks use proprietary protocols

and methods. In addition to this, the majority of these projects are closed source, relying on security through obscurity. Because of this, it is not possible to independently verify their security and there may be security issues present within the applications and frameworks. Through standardisation and open development (Google has attempted to start this with Google Nearby) secure development and validation of the product could be achieved. In addition to this, a standard would ensure that applications and frameworks would behave in an ethical and secure manner.

7 Practical Investigation

7.1 Tools

As the majority of portable electronic devices have built in speakers and microphones there is no requirement for specialised hardware to evaluate, transmit and receive near ultrasound signals. During our research, a number of different software packages were used to create tools to investigate the use of ultrasound as a communications channel. Although a variety of speakers and microphones were used for this research, the majority of work was performed using a laptop’s integrated speakers and microphone.

To visualise the ultrasound communications channel, an application such as Baudline or SDRSharp can be used. These software packages take a microphone input and analyse the spectrum in real-time. This is useful for capturing and analysing ultrasound data frames. It is especially useful for analysing unknown ultrasound signals as it allows quick, visual analysis in order to try to deduce signal characteristics such a modulation schemes and operating frequency.

GNURadio is a software defined radio package that can be used to perform signal processing in software. In addition to this, it also has functionality that can be used to encode and decode raw data. As a result, it is an extremely useful tool for creating transmitters and receivers and easily interfaces with any hardware such as microphones or speakers.

The identified use cases and previous research established the general characteristics of an ultrasound communications channel. The proof of concept aimed to mirror these characteristics and was developed to meet the following requirements:

- 18kHz – 22kHz range
- Frequency Shift Keying (FSK) modulation
- No more than 50bps
- Basic error correction
- No authentication

A general-purpose transmitter and receiver pair were developed using GNURadio. It is hoped that this will aid the analysis of future near-ultrasound systems by providing an easy to use framework for both transmitting and receiving near-ultrasound. The example below details the development of a 2FSK communications channel in the 18kHz-20kHz frequency range. Input is taken from a text file which is processed, encoded, modulated and sent to the speakers for transmission. At the receiving side, the microphone is used to record audio which is demodulated, decoded and processed. The received characters of the text file are then outputted to a terminal console.

7.2 Transmitter

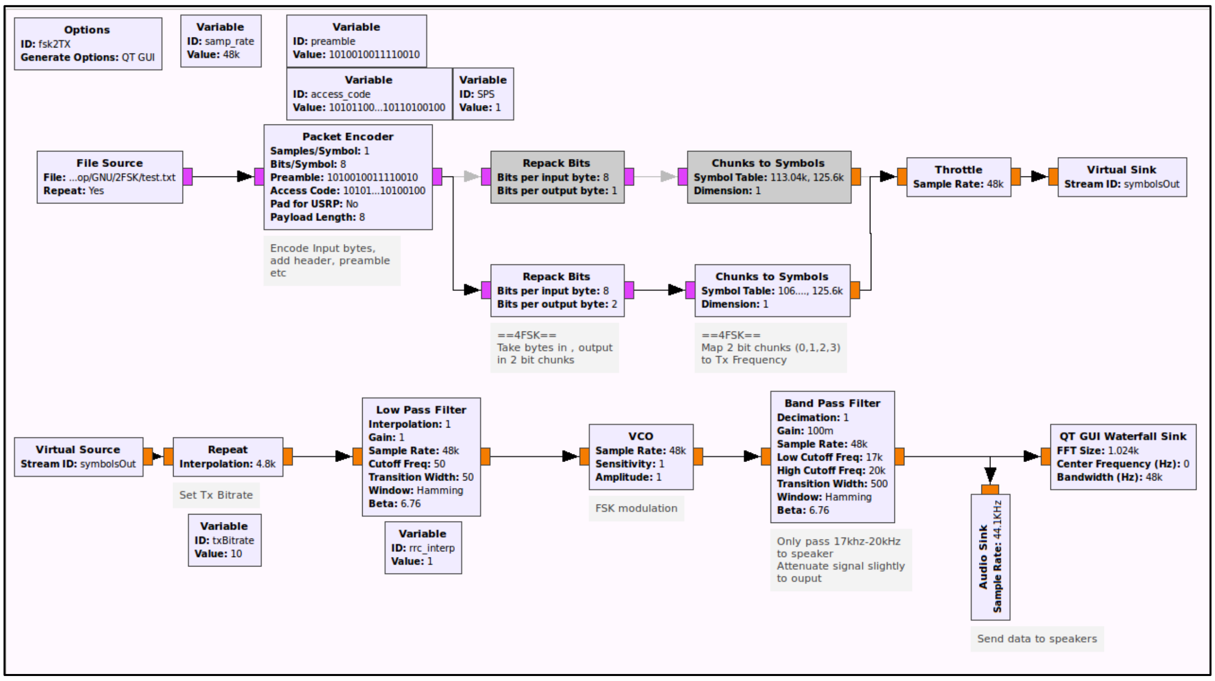


Figure 10 - GNURadio Ultrasound Transmitter

Source - The transmitter can have any source you wish, in the example below, a simple text file is set as the source. GNURadio will read in this text file byte by byte.

Packet Encoder - These bytes are then fed to a packet encoder. This packet encoder packs up “Payload Length” bytes into a packet and adds a preamble and access code to allow the receiver to sync its clock and correlate the incoming bits so that the payload can be extracted.

Repack Bits - The bytes are then repacked, taking the byte output from the encoder and outputting 1s and 0s.

Chunks to Symbols - These 1s and 0s are then mapped to their respective symbols which in this case is the frequency corresponding to 1 and 0.

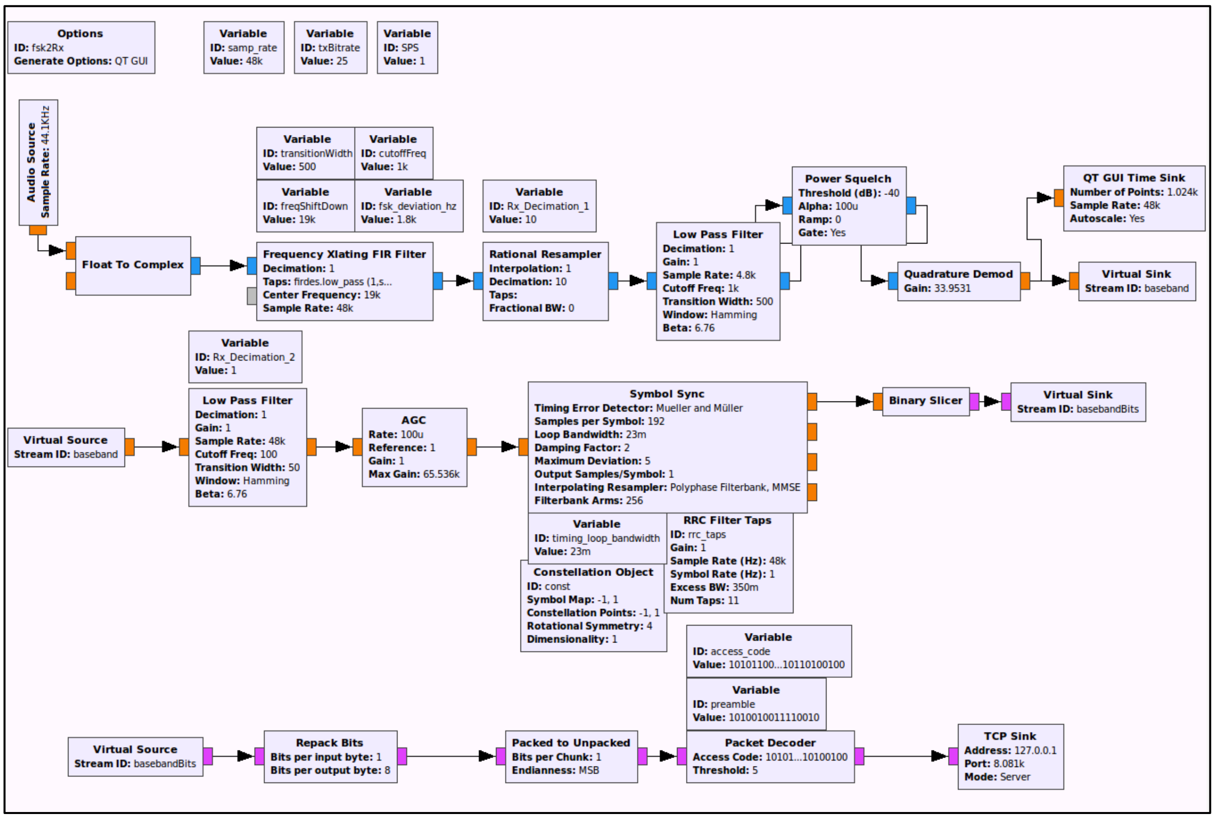
Repeat - The repeat block is used to repeat x samples for a single input sample, in this case this is used to set the transmission bitrate.

Low pass Filter - This baseband signal is then filtered to get rid of high frequency components.

VCO – This filtered signal is passed to a VCO block which maps the amplitude of the input signal, to the frequency of the output signal. This VCO block effectively acts as the FSK modulator.

Audio Sink - This FSK modulated signal is then passed to the computers speakers for transmission.

7.3 Receiver



Options ID: fsk2Rx Generate Options: QT GUI

Variable ID: samp_rate Value: 48k

Variable ID: txBitrate Value: 25

Variable ID: SPS Value: 1

Audio Source Sample Rate: 44.1KHz

Variable ID: transitionWidth Value: 500

Variable ID: cutoffFreq Value: 1k

Variable ID: freqShiftDown Value: 19k

Variable ID: fsk_deviation_hz Value: 1.8k

Variable ID: Rx_Decimation_1 Value: 10

Float To Complex

Frequency Xlating FIR Filter Decimation: 1 Taps: firdes.low_pass (1,s... Center Frequency: 19k Sample Rate: 48k

Rational Resampler Interpolation: 1 Decimation: 10 Taps: Fractional BW: 0

Low Pass Filter Decimation: 1 Gain: 1 Sample Rate: 4.8k Cutoff Freq: 1k Transition Width: 500 Window: Hamming Beta: 6.76

Power Squelch Threshold (dB): -40 Alpha: 100u Ramp: 0 Gate: Yes

Quadrature Demod Gain: 33.9531

QT GUI Time Sink Number of Points: 1.024k Sample Rate: 48k Autoscale: Yes

Virtual Sink Stream ID: baseband

Variable ID: Rx_Decimation_2 Value: 1

Virtual Source Stream ID: baseband

Low Pass Filter Decimation: 1 Gain: 1 Sample Rate: 48k Cutoff Freq: 100 Transition Width: 50 Window: Hamming Beta: 6.76

AGC Rate: 100u Reference: 1 Gain: 1 Max Gain: 65.536k

Symbol Sync Timing Error Detector: Mueller and Müller Samples per Symbol: 192 Loop Bandwidth: 23m Damping Factor: 2 Maximum Deviation: 5 Output Samples/Symbol: 1 Interpolating Resampler: Polyphase Filterbank, MMSE Filterbank Arms: 256

Binary Slicer

Virtual Sink Stream ID: basebandBits

Variable ID: timing_loop_bandwidth Value: 23m

RRC Filter Taps ID: rrc_taps Gain: 1 Sample Rate (Hz): 48k Symbol Rate (Hz): 1 Excess BW: 350m Num Taps: 11

Constellation Object ID: const Symbol Map: -1, 1 Constellation Points: -1, 1 Rotational Symmetry: 4 Dimensionality: 1

Variable ID: access_code Value: 10101100...10110100100

Variable ID: preamble Value: 1010010011110010

Virtual Source Stream ID: basebandBits

Repack Bits Bits per input byte: 1 Bits per output byte: 8

Packed to Unpacked Bits per Chunk: 1 Endianness: MSB

Packet Decoder Access Code: 10101...10100100 Threshold: 5

TCP Sink Address: 127.0.0.1 Port: 8.081k Mode: Server

Figure 11 - GNURadio Ultrasound Receiver

Audio Source – Takes audio input from the microphone.

Frequency Xlating FIR Filter - The signal is shifted down by 19kHz and passed through a lowpass filter. The down conversion is used to centre the signal so that the symbols (1 and 0) are represented by +1kHz and -1kHz respectively. The low pass filter is used for signal conditioning.

Rational Resampler – Reduces the numbers of samples to reduce the computational requirements and improve software performance.

Quadrature Demod – Converts the +1kHz and -1kHz bursts from the complex frequency space into one dimensional amplitude. i.e. from +1kHz -1kHz to +1000 and -1000 floats.

AGC – Automatic gain control ensures that the -1000 to +1000 values are mapped to -1 to +1 for processing by the symbol sync.

Symbol Sync – The output from the AGC consists of many samples which may only represent one bit. For example a bitstream of 1010 at the transmitter side when sampled may look like 111111111000000001111111100000000 depending on the sample rate. This block takes the many samples at the input and tries to lock onto the symbol rate from analysing the samples it is given. Once this has been established it outputs the value of the sample at the middle of each symbol period giving the original 1010.

Binary Slicer – The binary slicer makes a binary decision based on the input, if the input is less than 0 a 0 is outputted, if the input is greater than 0 a 1 is outputted. This block effectively outputs the original bits that were transmitted.

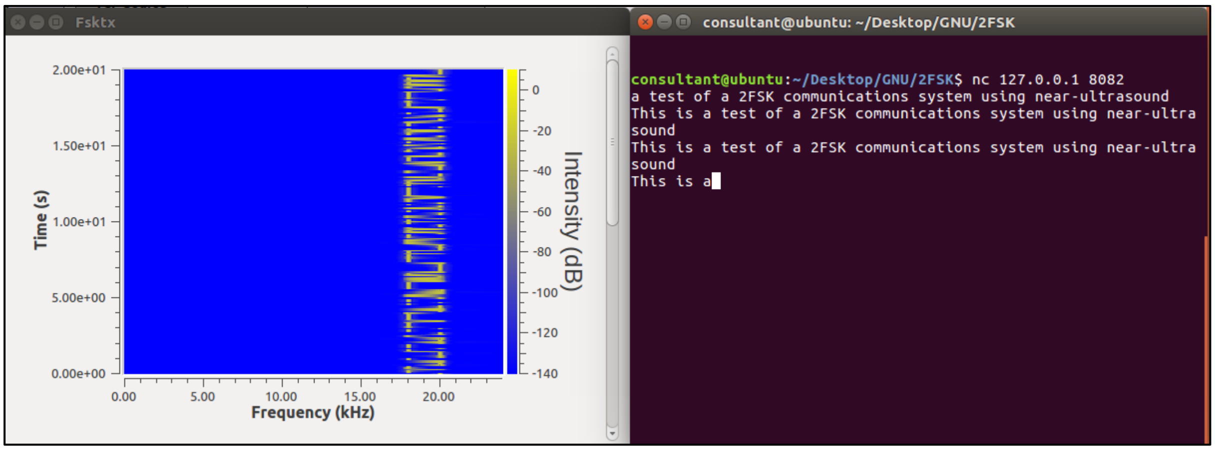
Repack Bits – This block takes in chunks of 8 bits and converts them to bytes.

Packed to unpacked – Convert back to bits with MSB endianness for the packet decoder.

Packet Decoder – This block takes the decoded bits as an input and looks for the access code that was added as a header by the packet encoder on the transmitter side. Once it has found this, it can frame the incoming packet and extract the payload. The threshold is used to specify how many errors can be present in the received access code for the packet decoder to assume that it has captured and framed the incoming access code and data. This block outputs the bytes that were originally encoded at the transmitter side. In this case this corresponds to the characters in the text file.

Sink - This data can be outputted in any way you want. In this case a TCP sink is used so that it is possible to see the data being received in real time. This TCP sink is accessed using netcat (*nc 127.0.0.1 8001)*

7.4 Results



```
consultant@ubuntu:~/Desktop/GNU/2FSK$ nc 127.0.0.1 8082
a test of a 2FSK communications system using near-ultrasound
This is a test of a 2FSK communications system using near-ultra
sound
This is a test of a 2FSK communications system using near-ultra
sound
This is a
```

Figure 12 - Screenshot Showing the Received Modulated Signal and Demodulated Text

Using the above example it was possible to achieve a data transmission of up to 50bps at a distance of up to two metres. A screenshot showing the received modulated signal and the demodulated and decoded text file is shown above.

8 References

[1] " https://blog.faberacoustical.com," Faber Acoustical. [Online].

[2] Y. S. A. D. Y. E. Mordechai Guri, "MOSQUITO: Covert Ultrasonic Transmissions between Two Air-Gapped Computers using Speaker-to-Speaker Communication," Ben-Gurion University of the Negev, Israel, 2018.

[3] F. T. Comission, "FTC Issues Warning Letters to App Developers Using 'Silverpush' Code," FTC, 2016.

[4] E. Q. C. W. a. K. R. Daniel Arp, "Privacy Threats through Ultrasonic Side Channels on Mobile Devices," Technische Universitat Braunschweig, Brunswick, 2017.

[5] V. Mavroudis, "On the Privacy and Security of the Ultrasound Ecosystem," *Proceedings on Privacy Enhancing Technologies,* 2017.

[6] L. C. Mathieu Cunche, "Analysis of an Ultrasound-Based Physical Tracking," 2018.

[7] E. Q. C. W. C. R. Daniel Arp, "Bat in the Mobile: A Study on Ultrasound Tracking," Technische Universität Braunschweig, 2016.

[8] O. B. H. S. R. a. R. H. Sverre Holm, "INDOORS DATA COMMUNICATIONS USING AIRBORNE ULTRASOUND," in *IEEE International Conference on Acoustics, Speech, Signal Processing*, Philadelphia, 2005.

[9] N. Pasham, "A noobs attempt to reverse engineer Google's cash Tex mode," Medium, [Online]. Available: https://medium.com/@nihal.pasham/a-noobs-attempt-at-reverse-engineering-google-s-cash-tez-mode-part-1-fa7f6d93320b. [Accessed 31 01 2019].

[10] Chris@twocentstudios.com, "Deconstructin a chirp.io," Two Cent Studios, [Online]. Available: http://twocentstudios.com/2012/10/12/deconstructing-a-chirp-dot-io/. [Accessed 31 01 2019].

[11] B. Schneier, "badBIOS," Schneier on Security, 4 November 2013. [Online]. Available: https://www.schneier.com/blog/archives/2013/11/badbios.html.

[12] Y. S. A. D. Y. E. Mordechai Guri, "DiskFiltration: Data Exfiltration from Speakerless Air-Gapped Computers via Covert Hard Drive Noise," 2016.

[13] Y. S. A. D. Y. E. Mordechai Guri, "Fansmitter: Acoustic Data Exfiltration from (Speakerless) Air-Gapped Computers," 2016.

[14] M. H. a. M. Goetz, "On Covert Acoustical Mesh Networks in Air," Wachtberg, 2014.

[15] D. Analusis, "©Disruptive Analysis Ltd, June2017Data over Sound Technology1Data over Sound Technology:Device-to-device communications & pairing without wireless radio networks," Disruptive Analysis, 2017.