TDDE21 - DRIP

Joakim Forsberg Oliver Johns Niklas Larsson
Hampus Runesson Mattias Salo

November 2020

Examiner: Andrei Gurtov

Contents

1 Introduction

This report will present the results of the project *Cryptographic Drone ID* made in the course *TDDE21 - Advanced Project: Secure Distributed and Embedded Systems*. The goal of the project was to create a prototype drone ID as specified by *DRIP IETF Working Group* [5], where the ID broadcasts over Bluetooth or WiFi in the form of a *HIP Host Identity TAG* (HIT). The project was performed by five students from Linköping University, the supervisor provided resources such as Raspberry Pis to make the project possible.

2 Background

In this chapter some necessary background information is presented.

2.1 Drone Remote Identification Protocol

The Drone Remote Identification Protocol (DRIP) is a protocol created by IETF Working Group to enable remote identification and tracking of unmanned aircraft systems (UAS). The purpose of the protocol is to provide safety and security by giving both civilians and law-enforcement a way to identify UAS.

DRIP specifies how a trustworthy remote ID can be produced and shared for others to identify. The identification of the drone is broadcast over WiFi or Bluetooth as and Host Identity Tag (HIT). The HIT tag should be broadcast in 20 bytes to make the protocol viable in areas where network and device bandwidth, processing power, and battery life are severely limited. Devices can thereby receive the drone ID in the form of a generated HIT tag and do a lookup against a database to both identify and fetch information about the particular drone [5].

2.2 Bluetooth advertising

With the release of Bluetooth 4 BLE (Bluetooth Low Energy) advertising was introduced. BLE advertising allows devices to broadcast advertising packets up to 32 bytes. This type of broadcasting also includes an RSSI value to allow for distance estimation. When Bluetooth 5 released, extensions to the LE advertising were added. Most notably, the range increased, and the advertising packet size increased from 32 bytes to 255 bytes [7].

2.3 WiFi

IEEE 802.11 is the collection of network standards that constitute the backbone of today's wireless network communication. Networks constructed accordingly are commonly referred to as WiFi networks [4]. These types of networks typically broadcast a Service Set Identifier (SSID) which can be used by devices to connect to the network. The SSID byte sequence can be used to broadcast any data.

3 Method

In this project, the goal was to create an implementation of the DRIP protocol using both WiFi and Bluetooth to broadcast a remote ID in the form of a HIT tag. The project started with a period of information gathering on the DRIP protocol and the other material given by our supervisor. When we understood how the protocol was supposed to work, we started the development. The development process focused initially on WiFi broadcasting, later on, however, the focus shifted towards Bluetooth broadcasting instead. With the use of Raspberry Pis substituting actual drones, development of DRIP broadcasting could begin. In parallel, an Android application began development to receive DRIP messages via WiFi and Bluetooth broadcasts. The application would be able to query information about the drone with the received HIT tag from a remote server. Below is a more detailed plan of what and in which order each task has been done.

1. Gathered relevant information in order to create a plan for the project
    - Read about DRIP
    - Gathered information from other sources supplied by our supervisor
    - Reviewed sample code from OpenDroneID [2]
    - Created a model of how the different parts worked together as can be seen in Figure 2
2. Development of Wi-Fi sender
    - Installed OS on the Raspberry Pis
    - Cloned the git repository from OpenDroneID and tried to get it to work
    - Contacted the author of the repository in order to get further understanding on how to get the sender part to work
    - Evaluated if we could get the Wi-Fi sender to work
3. Development of Bluetooth sender
    - Looked into different possible bluetooth beacon methods
    - Tried iBeacon, Eddystone and PyBeacon
    - Decided to use iBeacon in order to send the ID from the Raspberry Pi
4. Generation of HIT tag
    - Tried to get the HIT generation tool to work from the HIP repository
    - Downgraded the OpenSSL library in order for it to work
    - Generated the HIT tag

5. Complete the Bluetooth sender
    - Created a script that automatically generated a HIT tag
    - The script also automatically broadcast the HIT tag on startup
6. Development of receiver application
    - Looked into libraries that could be helpful for discovering Bluetooth broadcasts
    - Used AltBeacon in order to recieve the broadcasts [1]
    - Created a web server in order to store information about a drone ID
    - Implemented http requests in the application in order to get information from the server about a specific drone ID
    - Initialized Google Maps API in to enable tracking of the drone in the application

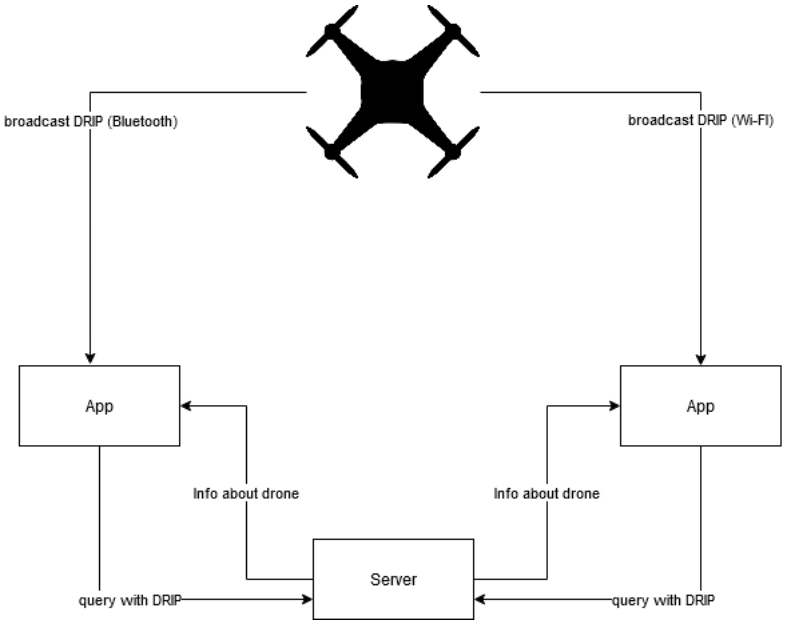


Figure 1: Initial model of the project

4 Result

The result of the project was a prototype of the DRIP with broadcasting over Bluetooth. For this prototype some bash scripts were created, an android application and a simple back end for the android application. The Scripts that were

created was a HIT gen script and the second one was for starting the Bluetooth broadcast and setting the payload for that broadcast. The android application can read the Bluetooth broadcast and read the HIT from the message. It can then use this HIT to do a search in the database for more information about the drone. In this information, there are some fields that indicate where the drone is located.

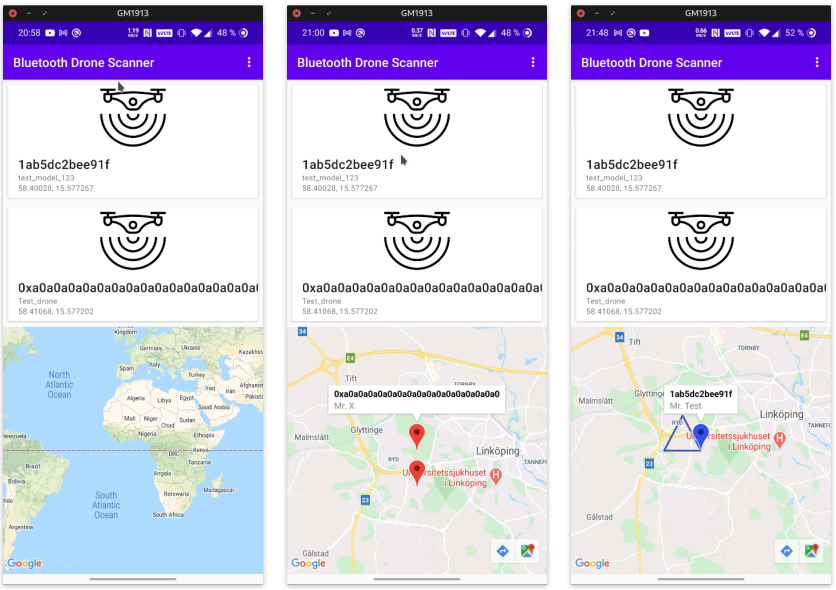
4.1 Bash script

The First script is supposed to generate a HIT tag for the Raspberry Pi if it doesn't have one. It does this by first downgrading the OpenSSL package to 1.0.2, when it has done this it clones the bitbucket repository OpenHip. After this it compiles the source code for OpenHip, this gives the file a hitgen file which the script executes. When hitgen executes, it requests a seed and then it writes the HIT and the different variables to a .xml file.

The second script starts the Bluetooth broadcast by using the hcitool. It takes the HIT as input in the format of 2001:15:76a9:34cc:a0bb:cf1b:63a7:b712. It then takes this HIT and modifies this HIT to look like
20 01 15 76 $a9$ 34 cc $a0$ bb cf $1b$ 63 $a7$ $b7$ 12. Sets the payload of the broadcast message to this HIT and then starts the broadcast with and broadcast interval of 1 sec.

4.2 Receiver Android Application

The Android application's primary purpose is to receive Bluetooth broadcasts from drones. It utilizes the AltBeacon[1] Android package, licensed under the Apache-2.0 license, to receive BLE broadcasts. It identifies drones by matching patterns to the detected broadcasts, which might be sending out drone-IDs. These broadcasts are used to query a web server for additional information about the drones. If the web server provides information about one or more drones, this will be shown in the app in a list along with markers on a map and a path where the drones have moved. The map requires a Google Maps API key. The application is not capable of receiving WiFi broadcasts from drones as originally planned.



(a) List of drones (b) Marker information (c) Path of drone

Figure 2: Collection of Android application screenshots

The application is written in Java and built using Gradle; the application is using the AltBeacon library to receive Bluetooth broadcasts, the Google Maps API to show a map, and OkHttp to make HTTP requests. Since the application is written in Java and is using object-oriented programming, changing the Bluetooth receiver module of the application could be done with minimal modification to the program as a whole.

4.3 Web server

The web server is a simple docker-compose system consisting of a Python Flask container, a MySQL container, and a PHPMyAdmin container. The Flask container hosts a server with a REST API with support for several different requests, such as adding drones or getting drone information. The MySQL container provides the system with a database containing all the drones, and the PHPMyAdmin container serves as a graphical interface for observing and changing the database. Currently, the database saves information about:

- ID
- DRIP ID
- Timestamp
- Model of aircraft

- Latitude
- Longitude
- Owner

5 Discussion

5.1 WiFi

As mentioned in the chapter about the method, the first approach taken was to try to use WiFi to broadcast the HIT. According to the DRIP protocol this should be done by using Neighborhood Area Network (NaN). As mentioned in chapter 3 there was a some sample code in the OpenDroneID repository that will broadcast a Remote ID over Bluetooth or WiFi-NaN, so in the beginning some work was done to try to get this repository to work, and if that works the idea was that then the payload of the messages could be swapped to the HIT. Unfortunately we weren't able to get this repository to work as intended. The code compiled and could be started on a device, the problem was that there was no broadcast happening even when it had been started. Some different methods was tried to see if the broadcast was started. Most of the methods that was tried involved some sniffing on the network with the help of wireshark. As mentioned also in chapter 3 we contacted the author of the repository and ask how to test the code and sniff some network packages from the broadcast. The answer that was given was to start the broadcast on the raspberry pi, and also convert the raspberry pi to an access point, then connect to that access point with another device and then sniff the network on that device. Unfortunately this didn't give any results either. This was one of the reasons that a switch to Bluetooth was made.

5.2 Bluetooth

With the addition of LE advertising in Bluetooth 4 broadcasting turned out to be much easier then first suspected. There were much information already available on how the low level Bluetooth controller commands worked. Since all three standards, iBeacon, AltBeacon and Eddystone, are so similar in structure the application could be very easy to convert to any standard should the need arise. The standards could even be dropped in it's entirety since all they do is determine a structure to followed. Although for this project having a predetermined structure helps with app development. When broadcasting was working as intended we moved onto changing the payload. The HIT is according to it's definition 16 bytes which fits well with the available payload in all three broadcasting standards. There was however a part missing. A HIT is not secure, as in you can not ensure that the owner sent the HIT with an attached signature. This is covered in the RFC for the DRIP protocol and there it is stated that the signature should be 64 bytes (using ECDSA). As previously mentioned the

available payload is only 16 bytes with LE advertising on Bluetooth 4. However, with the introduction of Bluetooth 5 that advertising has been extended to 255 bytes. We had a Raspberry Pi 4 available and expected to be able to use its new Bluetooth 5 ready controller. Unfortunately the "new" Bluetooth controller is in fact not new. It's the same one used in the Raspberry Pi 3b+ with an extended firmware to support some of the new features of Bluetooth 5. The extended advertising packets was not included in this extension of the firmware. This meant the Raspberry Pi 4 needs a third party Bluetooth 5 sender/receiver. When this was discovered there was to little time left in the course to get the needed device and thus more focus was instead aimed at the finalizing what we had.

5.3 Android Application

The application in its current state captures Bluetooth broadcasting with the AltBeacon API, which uses Bluetooth 4 Low Energy. Bluetooth 4 might be a bad fit for sending drone IDs since the range could be short for Bluetooth 4 Low Energy applications. However, this is a limitation of the current built-in Bluetooth hardware of the Raspberry Pi. If the drone IDs were sent out using Bluetooth 5, a higher strength, and thus hopefully providing a longer range, the functionality of the system might closer match the aim. The support of this for mobile handsets using the Android operating system started appearing around 2017, with the Samsung flagship, the Samsung Galaxy S8 being the first to support the standard [6]. The handsets are also not required to support all the broadcasting standards for Bluetooth 5 [8], which might cause compatibility errors for phones of different manufacturers where not every phone supports all the features of the Bluetooth 5 standard. This might pose a problem in application development if phones that don't support all features should be catered to as well with differing solutions.

Further, the communication between application and web server is now not using encryption. This was allowed since we wanted something fast and something that worked with minimal development time and a solution that was only used when testing functionality internally.

6 Future work

6.1 WiFi broadcasting

The WiFi solution still has a lot of work to be done to resemble the proposed DRIP implementation. As mentioned in the discussion, the open drone id project lacks some features that are important for DRIP implementation. Most notably, the lack of real broadcasting. Because of this, we think that future work relating to the WiFi implementation should focus on building and implementing support for the NaN protocol to enable broadcasting over WiFi. We would also suggest that a development kit for WiFi is used for such a task. This, to ensure

that the needed functionality is available, something that can be a problem on devices such as the Raspberry Pi.

Another area of great interest to the WiFi implementation is an Android/iOS app that also supports the NaN protocol. To make a complete solution supporting both Bluetooth and WiFi. The new functionality could be added to our Android app since it already supports Bluetooth broadcasting drones. The Android operating system has some support for WiFi NaN, in Android called WiFi Aware [3].

6.2 Bluetooth broadcasting

As we mentioned in the discussion, the Raspberry Pi did not have full support for some of the functionality that Bluetooth 5 adds. Specifically a broadcast of 256 bytes instead of the 32 bytes that Bluetooth 4 supports. Thus, we suggest that some future work could be directed towards working with modules/development kits that support the full functionality of Bluetooth 5. The HIT/HHIT is 16 bytes large, and the signature is 64 bytes large. Using full Bluetooth 5 functionality would mean that there is plenty of space in the extended broadcast message format to send the identity with its signature.

7 Conclusion

Working with the Drone Remote ID Protocol has been both fun and interesting. We spent alot of time researching and trying to get the Wi-Fi implementation of DRIP to work. Unfortunately, we bumped into alot of problems along the way and never found a good solution. Instead, we switched focus to the Bluetooth implementation, which we found easier to implement and did not suffer from the same problems as the Wi-Fi implementation.

In summary, our goal was to create both a Wi-Fi and a Bluetooth implementation, which we were unable to fulfill. However, we have found the difficulties regarding the Wi-Fi implementation and created a working Bluetooth prototype which we, in the end, are satisfied with.

References

[1] Altbeacon. https://github.com/AltBeacon/android-beacon-library. Accessed: 2020-12-07.

[2] Open drone id. https://www.opendroneid.org/code/. Accessed: 2020-12-03.

[3] Wi-fi aware overview. https://developer.android.com/guide/topics/connectivity/wifi-aware. Accessed: 2020-12-09.

[4] Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, 2016.

[5] DRIP IETF Working Group. *Drone Remote ID Protocol (drip)*. IETF.

[6] John Leonard. Bluetooth 5 in smartphones. https://blog.nordicsemi.com/getconnected/bluetooth-5-in-smartphones, Jun 2018. Accessed: 2020-12-09.

[7] Daniel Murfet. Foundations for category theory. https://www.bluetooth.com/bluetooth-resources/bluetooth-5-go-faster-go-further/.

[8] Jack Price. Not all bluetooth 5-enabled smartphones are created equally. https://www.xda-developers.com/check-bluetooth-5-all-features-supported/, Mar 2019. Accessed: 2020-12-09.