TDDE21 *DRIP* 2024

Authentication Formats and Protocols for Broadcast Remote Identification

Jiajun Chen (jiach613)
Fabio Crugnola (fabcr549)
Albin Svärd Gruvell (albsv335)
Liza Johansson (lizjo663)

December 17, 2024

Contents

1 Introduction

This report presents the work made by four students in the course Advanced Project Course: Secure Distributed and Embedded Systems during the fall of 2024. This project is a continuation on what has been made by other students during the same course in previous years.

1.1 Motivation

As the number of drones in society increases it leads to an increase of harmful drones as well, where drones can be used to for example spy on individuals, military objects, or other malicious intents. In order to be able to penalize people using drones in a harmful way there needs to be some way for authorities to know who the drone belongs to despite the operator being far away. This is the purpose of *DRIP* (Drone Remote Identification Protocol), as it implements a verification process for drones where each drone is registered to an owner that can be identified by scanning for the drone with an app.

1.2 Project goals

The project goals for this year were:

- **Test DNS registry (registries-20) LIU-IDA/LITH-EX-A–24/073–SE.**
- **Auth formats to RFC9575.**
- **Test BT5 on RaspberryPI 5.**
- **Update Android application for the backend.**

2 Background

In this section, the background of the project is presented.

2.1 Drone Remote Identification Protocol

The Drone Remote Identification Protocol (*DRIP*) is a product of the IETF (Internet Engineering Task Force). This protocol brings up how trust policies and periodic accesses to registries change how UAS (Unmanned Aircraft System) and their Remote Identification (RID) can be evaluated in real-time, enhancing their trustworthiness. In this system, each drone and each operator have their own unique identifier linking them together, reassuring accountability for operators and their respective drones. This protocol can be used with technologies such as WiFi and Bluetooth 5.[1]

2.2 *DRIP* Identity Management Entities (DIMEs)

DRIP Identity Management Entites(DIMEs) include a list of entities that follow a hierarchy structure. In this structure, the entity at a higher hierarchy is responsible for endorsing the entity one layer below it. The endorsement process occurs once during the public key creation period for each entity[2] and is also included in the *DRIP* link message each time the broadcast endorsements are sent[1]. In this section, we briefly outline the process of endorsement during public key creation. We further explain how this hierarchy is represented in DNS and during broadcast endorsement in 4.3.1 and 4.4.1 respectively.

In this project, the RAA with ID 16376 is allocated to the *DRIP* working group, which is responsible for endorsing the HDAs it manages. Linköping University has been assigned the HDA with ID 1026 for testing purposes.

The following steps outline the process for the authorized entity RAA (16376) to endorse the unauthorized entity HDA (1026):"

1. The project team(LiU) for HDA(1026) generate DET for Authorization purpose. This step also generate the Public and Private key pair for this given DET as well as the CSR(Certificate signing request).
2. The project team(LiU) for HDA(1026) send the CSR to upper entities that are responsible for the endorsement of their DET. In this case, we send CSR to RAA(16376)
3. The management team of RAA(16376) use their authorized RAA endorsed/signed CSR with their private key and send back a signed certificate.
4. We registered the authorized HDA's DET to our DNS server along with its certificate.

After the process above, our authorization HDA are endorsed and in charge of the authorization for other entities below it. We can use our endorsed authorization HDA to endorse the issue HDA we generated and further endorse all our drones using our issue HDA by following the same steps.

Noted that the endorsements generated from our year might have been already expired by the start of 2025. Also due to the fact that the *DRIP* scripts had some minor bugs [3], and the scripts might already be updated in 2025, these whole steps of endorsement might have been performed again for next year.

2.3 Bluetooth

Bluetooth technology enables short-range wireless communication, offering a range of features to facilitate device interaction. One such feature is Bluetooth advertising, which allows devices to broadcast

information to nearby devices in the local area. Starting with Bluetooth 4, a Low Energy (LE) mode was introduced, making it ideal for battery-powered devices. Bluetooth 5 expanded on this by introducing new encoding modes, including the coded PHY mode, which supports communication over longer distances (up to one kilometer) and larger data packets (up to 255 bytes) [4].

In Linux, the Bluetooth stack is implemented as BlueZ, the official kernel-level protocol stack for Bluetooth. To interact with BlueZ, the Bluetooth Host Controller Interface (HCI) is used. A popular tool for interacting with Bluetooth controllers in Linux is hcitool, which enables users to issue HCI commands directly. The types of commands that can be sent, along with their functionality, are detailed in the Bluetooth Core Specification [5].

2.4 Wifi Aware

WiFi NAN (Neighbor Awareness Networking), also known as WiFi Aware is a feature introduced in modern Wi-Fi standards. It enables nearby devices to discover and communicate with each other directly without requiring a traditional Wi-Fi network or internet connection. This technology is designed to support proximity-based services and works well for applications that require contextual awareness [6].

It is one of the official mean of communications for *DRIP* and ASTM Messages.

2.5 Previous work

The work done by last year's group consisted of the implementation of the DNS for *DRIP*. This was implemented similarly to the current implementation, however, the record type was not of the type described in [7]. Furthermore, the 2023 working group also implemented the authentication messages for *DRIP* (Link, Wrapper, and Manifest), which were adapted for Bluetooth 4. This implementation was also extended to the observer app which was made to support the *DRIP* framework for Bluetooth 4. [8]

3 Method

The concrete implementation of *DRIP* in 2023 consisted of a series of showcase examples involving the main components described in section 2. Specifically, the *DRIP* examples could/can be organized in the following modules:

- ***DRIP* Transmitter:** A basic Python implementation of a *DRIP* Transmitter able to send *BASIC ID ASTM* messages.
- ***DRIP* Observer:** A fork of the *opendroneid observer* [9] Android app that was modified to support *DRIP* in the past years on top of the *ASTM* protocol.
- **DNS Server:** A DNS server implementation to support validation and identification of *UAS*.
- **Entitites DETs generation scripts:** Some utility scripts provided by the *IETF DRIP Remote ID protocol group* [10].

Given that the totality of the deliverables consist of source code supporting *DRIP* features, a great amount of time was initially spent to reorganize the repository containing the various modules. Up to this point, it was not well organized. The modules were indeed part of a single repository, thus changes to the various modules were interleaved in a single git history, making it extremely hard to unroll bad changes, merging branches and more.

For this year, a better approach was chosen and each module now has a separate repository. To keep source code forks (such as the Observer) up to date with the original repositories, mirror repositories have been set up to keep track of changes easily. The new repositories have been forked from the mirrors, and the old changes made by the other groups have been reintegrated.

The approach used to tackle the problem is inspired by what the previous group working on this project did [8]. After getting a thorough understanding of the material mentioned in Section 2, a team of two people worked on the changes needed in the *observer* module, while the remaining two worked each on the *transmitter(s)* and the *DNS Server*. In the second part of the semester we decided to reassign some of us where most of the help was needed. Communication between the various team members was held through *Teams* chat-group and a weekly meeting.

The new organization, helped various member using project management features that GitLab offers such as issues management, pipeline test's integration and code review.

3.1 Tasks

The goals mentioned in section 1 materialized in more specific tasks, which have been grouped below by the same goal.

3.1.1 Auth formats to RFC9575

This goal consisted of two tasks: update the *observer* and *transmitter* with changes required by *RFC9575* draft. The Android *observer* will not see major reorganizations, but a set of patches to adapt it to RFC9575. On the other hand, for the *transmitter* a more radical approach was taken.

Following the suggestion from 2023's team [8], the Python based transmitter was scrapped for a fork of the *opendroneid ASTM* transmitter. This transmitter comes with the advantage of having the entire *ASTM* protocol already implemented and tested, based on the *opendrone-core-c* library [11]. Moreover, the C Programming language adapts better to the kind of hardware which usually consists of microcontrollers installed on *UAV*s.

3.1.2 Test BT5 on RaspberryPI 5

The need for Bluetooth 5 testing, in the context of *DRIP* comes from the presence in the Bluetooth 5 standard of what is known as *long range transmission* [4]. Long range advertising makes it possible for Bluetooth transmitters to advertise packets at really slow speed for *a couple* of kilometers. This feature is extremely useful when it comes to *UAV*s flying in the range of an *observer*.

Long range transmission is a non-compulsory trait of Bluetooth 5 chips. Sadly after some research, it was realized that the Raspberry Pi 5, although being Bluetooth 5 certified, does not support *extended advertising*.

Following these discoveries, other hardware was given to the group to make testing possible: two NRF52840 Development Kits [12] in the form of a Dev Board and USB Dongle Chip. Moreover, an ESP32-S3 Dev Kit by Waveshare [13], was also given to the group to test another feature supported by the *observer*: WiFiNAN transmission.

Another device that was given to the group is a Crazyflie drone [14]. These drones have an open-source reprogrammable firmware. They are controlled by an STM32 Chip which handles the flying control loop. It also comes with a NRF51x chip, used to provide Bluetooth and Proprietary transmission for controlling the drone remotely (through a smartphone or a proprietary remote device). A task given here was to have *DRIP* transmitter implementation running on the drone hardware, in order to be able to make more realistic tests.

The introduction of new hardware, came with the need of having more adaptable software, especially in the context. The decision of scraping the Python-based transmitter for a C based one, became even more beneficial: a lot of code could be shared among the various transmitters.

3.1.3 Update DNS registry (registries-20)

The primary goal contains implementing a prototype DNS server based on the suggestion from 2023's teams [8] and update its implementation to include the latest changes from draft drip-reg-20[7]. The main focus here, is taking advantage of the newly defined resource record type used in *DRIP*, as well as the DET Hierarchy into the DNS. Other aspects, such as security consideration and DNS operations, are less emphasized but are mentioned in the Future Work section 6 as suggestions for next year's team.

3.1.4 Update Android application for backend

This goal involves updating the *Observer* to comply with the RFC9575 standard, as well as updating the functionality for sending DNS queries.

4 Result

This section presents the results of the project.

4.1 drip-core-c library

As mentioned in Section 3, the various transmitters have many common source code parts. For this reason, a C library loosely based on the *opendroneid* implementation of the ASTM protocol *opendroneid-core-c* [11], was realized for *DRIP*. The result is *drip-core-c*.

The library is an extension of *opendroneid-core-c* and uses ASTM Authentication messages to craft *DRIP* encoded packets. It supports all currently defined *DRIP* messages, and can easily be integrated on different architectures and different cryptographic libraries.

4.2 Three transmitter

The library from the previous section has been successfully integrated in three different transmitter implementations:

- ***DRIP* Linux Transmitter:** This implementation replaces the Python based version. It supports all ASTM messages and is able to transmit WiFi beacons, Bluetooth 4, and Bluetooth 5 extended advertisements.
- ***DRIP* NRF52840 DK Transmitter:** This implementation uses Zephyr OS and the nRF SDK to implement a *DRIP* Transmitter. It supports BT4 Legacy Advertisements and Bluetooth 5 Extended Advertisement. Given the lack of support in the Linux Bluetooth Stack (bluez) for Bluetooth 5 extended advertisement, this transmitter implementation was used to test Bluetooth 5 with the Observer.
- ***DRIP* STM32-S3 Transmitter** This implementation is implemented with EspressIF SDK, and supports Bluetooth 4 and Bluetooth 5. It has never been tested extensively, due to the fact that was mainly conceived to work with WiFi NAN, which sadly is still not supported by the official SDK.

4.3 Observer up to RFC9575 standard

This section presents the major changes from auth-41 [15] to RFC9575 [1].

4.3.1 Receiving and verifying chain of endorsements in *DRIP* Link

We implemented the chain of endorsement of *DRIP* entities following RFC9575 section 6.3.

In the transmitter side, we should send all these four Broadcast Endorsement messages in *DRIP* Link. This is further confirmed in drip-dki-03[2] where it points out "It(The separation role of *DRIP* entities) does make the chain of trust for a HDA customers' Operational DETs to be 4 Endorsements."

1. BE: APEX, RAA
2. BE: RAA, auth HDA
3. BE: auth HDA, issue HDA

4. BE: issue HDA, UA

However, in our project, due to the fact that there is no APEX in place and the public key of RAA(16376) is still unknown, only 3 and 4 are sent from *transmitter*.

In *Observer*, we enqueue the BEs until all of them are received and verify the signature of each using the public key obtained from the BE immediately above it. The signature of BE from the highest level of the hierarchy needs to be verified by the public key that is cached in the *observer* app.

4.4 DNS

Based on the suggestion from 2023's team, BIND9 DNS server has been used. In order to make the configuration more accessible, we use ubuntu/bind9 docker image and make all server configurations visible in our GitLab repository.

4.4.1 DIME DET Hierachy

We stored the DETs that mangaed by HDA(1026) in a DNS zone file with the nibble reversing of the IPv6 address as per drip-reg-20 section 3[7]. The zone file can be found in our repository.

4.4.2 New resource record type

Compared to last year, HHIT and BRID resource record types have been introduced in drip-reg-20[7]. Currently, we use TYPE65280 as an undefined RR type since BIND9 does not inherently support adding fully custom RR types outside the standard DNS protocol. We only implemented HHIT resource record to provide a proof of concept. As per drip-reg-20, the HHIT Data should be encoded in CBOR bytes. The encoding script for HHIT data could be found at our DNS repository, while the decoding script could be found at the observer app by checking the DNS query related function. However, due to not using HHIT resource record type as defined in drip-reg-20 and lack of time, certain requirements are not met so far. First, the current text representation is in a string of hexadecimal digits instead of base64 (section 5.1.1 in drip-reg-20). The reason why hexadecimal digits is used for now is because this is the standard text presentation for unknown RR type as defined in RFC3597 section 5. Secondly, the registration-cert field was left empty since its length definition is still not clear (section 5.1.2 in drip-reg-20). Please see future work for more information regarding these new resource record types.

4.4.3 DNS operation

In drip-reg-20 section 7.1, a list of common practices in DNS operations is provided. In our work, DNS Dynamic Update[16] has been used. Specifically, user interaction to add, update, and remove resource records in the zone file is done by using nsupdate. The nsupdate[17] is a client command that allows the user to send dynamic DNS update requests to the DNS name server. Additionally, TSIG authentication is used for message exchanged between client and server. Please see future work for more information regarding the improvement of DNS operations.

4.5 Test result of Long range Bluetooth 5

Bluetooth 5 extended advertising is said to have a range of up to a kilometer in perfect conditions, compared to Bluetooth 4 which has a range of up to 100 meters [18]. A test was conducted to estimate the transmitter range under Bluetooth 5 Extended Advertising. The result of this test showed that the observer would receive data up to a range of 313 meters. The tests were carried out outdoors, in an environment with few obstacles that might have influenced the final results. Although the maximum technical range is 1 KM, in normal conditions it should be possible to easily reach 400-500 meters. We therefore consider the results of our tests acceptable.

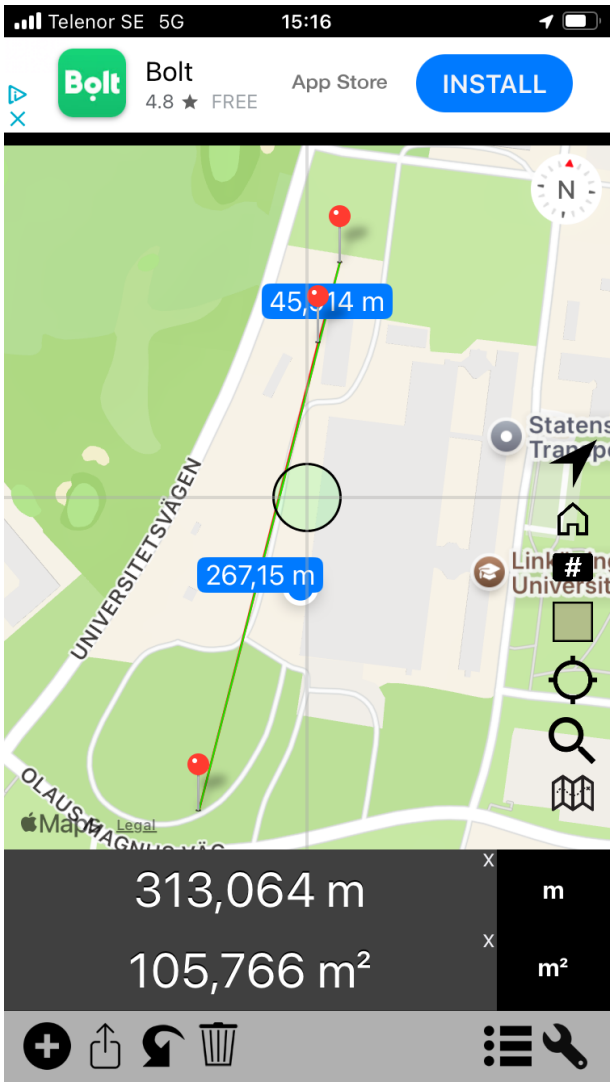

Figure 1: Measurement tests, in the picture the furthest point with signal and the transmitter fixed point

5 Discussion

In this section some challenges that were encountered are presented as well as some advice for future students continuing the work on this project.

5.1 Challenges

- Understanding the drafts took some time. We did not really understand them in the beginning despite reading through them several times. It took reading them multiple times during the project to get a better understanding of them.
- From previous years there is now a quite large code base with code from different sources and written by students from different years. This led to a large code base that took some time to understand. At the beginning of the project, it took some time to know where to begin and in which part of the code to develop our part in.
- There where also some difficulties with the required hardware, as we had to try multiple different hardware components before finding one that was compatible with the Bluetooth 5 extended advertising that we needed.

5.2 Advice for future students

- Read through the draft early in order to grasp the overall structure and purpose of *DRIP*. When doing this it can be good to have a document to keep track of the acronyms used in the draft and add a describing sentence for each of the acronyms. This both makes it easier to understand the drafts in the beginning and can also be helpful later in the project as some of the acronyms might only be used rarely during the project.
- As this is a new subject there is not that much info on the topic when looking for information on e.g. Google. So make sure to ask the assistant questions and discuss with them if things are unclear.
- Another advice would be having a structured way to share information with everyone in the group. There are many different parts of the project so everyone won't know everything about all parts of it. So having a structured way of sharing the gained knowledge so everyone is kept up to date can be good. This can be done through weekly group meetings, another tip is to have a document with all the relevant acronyms and relevant words so that everyone in the group can keep up and talk about the same things.

6 Future Work

In this section some points about the various modules and the necessities that still need to be addressed are listed.

6.1 Transmitter(s)

- The new transmitters support ASTM and the latest *DRIP* drafts. Unfortunately, it was not possible to test WiFi NAN, due to the fact that the WiFi NAN APIs from the SDK do not yet support the ESP32-S3 [19] (which should in any case support WiFi NAN at the hardware level). WiFi NAN is also a quite rare feature on modern smartphones which made testing practically impossible. More work to research WiFi NAN compatible hardware needs to be done. This implementation has been left as a consequence, as an afterthought, and features like BT4 and BT5 were tested with the NRF implementation. The ESP-32 source code can probably be scrapped, if next year WiFi NAN will be tested with another platform.

- The Linux transmitter is able to transmit Bluetooth 5 messages, but after a lot of research, it has been concluded that the BlueZ Linux Bluetooth Stack doesn't fully support extended advertisements (as of December 2024). Although messages can be clearly scanned with the command *btmon*, it seems that messages that are being sent on CODED PHY Primary and Secondary Channel are not correctly forwarded to the higher layer of the stack and then to the user application [20] [21]. Few resources and many complaints can be found on the Internet and hopefully, by the time this project is extended next year some improvements to the Bluetooth Stack will have been made. Until that moment, Bluetooth 5 can be tested with the NRF Transmitter implementation. Another thing that can be done to improve testing on the NRF transmitter is taking advantage of the LEDs and Buttons the board provides to have a way to easily switch from Bluetooth Legacy to Bluetooth 5 Extended transmission.

- The transmitters *main loop* could and should be improved. An interrupt based approach or a better message scheduling logic (based perchance on lookup tables or such), adhering to *RFC9575* must be implemented. Currently, the transmitters are not fully observing the message transmission frequency and delays mandated by the draft.

- The micro-controller based transmitters have the disadvantage of not having a way to synchronize time (for example through the internet), which is needed to have valid *DRIP* message. Currently, a hacky solution is being used, consisting in setting the micro-controllers timestamps to the firmware's compilation time. This works during rapid testing and prototyping (where the firmware gets flashed and compiled multiple times), but doesn't work in a more "production based" context. A solution, given that the transmission happens through Bluetooth could be to use the Time Sync Bluetooth Service (this solution is good for the Crazyflie), available on many smartphone to synchronize the micro-controller timestamp at boot with the one coming from a smartphone. Another simple and more robust solution could be to have a UART console where the user could just input the current time.

- Some initial analysis on the Crazyflie firmware were made, but no concrete implementation or tests have been done. The source code with some initial suggestions is available in the Crazyflie GitHub repository. The main obstacle to the implementation of *DRIP* is that the NRF SDK version being used on the firmware is quite old and doesn't match the one used for the NRF52840 transmitter. Moreover, it seems that the firmware according to some discussion happening under the original implementation disable Bluetooth advertising to improve performance [22] [23].

- More rigorous validations should be done to test thoroughly the Extended Advertising Capabilities of Bluetooth 5 on Long Range. For example, tests where the transmission power of the transmitter is changed could be made. [4] [18] [24].

6.2 Observer

In the year 2023, many issues regarding the implementation of the observer were highlighted [8]. With high certainty, this year some additional tech debt might have been introduced by supporting the additional features required by *RFC9575*. Among the issues listed in 2023, only one still hasn't been tackled, because most of the group focus was put on delivering *RFC9575* support:

- If a verified message is currently stored and shown in the interface, and a non verified message is received, the current implementation of the app will replace the message. An alternative to this could be to store the non verified message (to be able to check the hash contained in a manifest later), but keep showing the verified one on the interface until the newer message can be also verified.

The new changes brought to comply with RFC9575, brought to light some issues that will require a somewhat high amount of effort to be fixed.

- The current FEC implementation, only used with Bluetooth Legacy, only works in case the page that is lost is not *page zero* (or the last one, but that is a case not covered by *DRIP* in general). If the first page is missing, the observer should still be able to reconstruct it, but for that to happen, a new data flow architecture should be developed. The current implementation of the Observer was designed to receive general ASTM messages and not specific *DRIP* Messages.

  The Bluetooth Advertisement message counter sent by the transmitter is ignored during the reception and parsing of messages. The observer uses the ASTM last Authentication Page Index field to know whether we "finished" receiving all the pages of a message so that it is possible to proceed with parsing. If the first page is missing, this field is lost and there is no way for the *Observer* to know whether the transmitter has started transmitting a new message or not, because it does not know how many pages the current message consists of (this information is stored in *page zero*) [25].

  This issue highlights also the fact that if the Observer loses more than one page, of which one is *page zero*, then it might combine pages coming from two different ASTM Auth messages, that will get discarded later in the parsing process because corrupted. This is also a problem because in this way two messages get lost and the *Observer* will have to wait for retransmission to have the chance of parsing them.

  A solution to these problems could be using the Bluetooth Message counter that is currently sent (as per Bluetooth standard) and ignored to distinguish between the messages that are sent. The counter is set to 0 every time the transmission of pages coming from a new ASTM Auth message is started. In this way, every time the counter becomes lower than it was during the previous page reception, the *Observer* can assume that a new message is being transmitted and it can try to reconstruct a missing page (*page zero* included) if any.

- The Hash Verification for *DRIP* Manifest message [1] is not working consistently. The problem seems to arise because the same message, already parsed and verified is received at the observer side more than once, due to the way the transmitter retransmit advertisements. To avoid memory leaks, the observer deletes the most recent "verified" hash. There are some hardware level filtering options that can be set in the observer configuration for scanning messages, but they did not work during our troubleshooting. More work should be put here to find a good retransmission interval at the transmitter side and filtering options at the observer to avoid the hash verification failing. When the interval is too low duplicate messages are retransmitted, when the interval is to big then some messages are "lost", breaking the verification chain as well. Another robust solution could be to to just store the hashes in the cache for longer time and remove them after they are "hit" for a predefined amount of time, but this comes with increased memory usage and might also lead to memory leaks in the long run.

Under the *opendroneid* GitHub webpage [11] there is also another version of the observer, based on a multi-platform framework. It would be nice to investigate how long the implementation of *DRIP* would take there. The result would lead to the ability to support other platforms, such as iOS at *observer* side.

6.3 Wireshark detector

It would be nice to extend the Wireshark filter made by *opendroneid* group [11] in order for it to support *DRIP* messages as well.

6.4 DNS

- In RFC9575[1] section 3.1.1 , they mentioned the verification of *DRIP* with internet access as follow

  "The Observer MUST perform a DNS query, when connectivity allows, to obtain a previously unknown HI."

  However, our investigation, including a review of draft-drip-reg20[7], did not clarify how to expose the HI/public key in DNS.

- Update to a new drip-reg draft or RFC standard by the time you see this report. One thing could be implementing the new resource record types so that it can be recognized by the DNS server. A possible implementation to add a new type of recourse record required changing the source code for BIND9. We forked the official BIND9 repository into our workspace and added a Dockerfile to build and use it easily. The direction of implementation can be found here[26].

- Improve user interaction to manage the DNS server. Currently, a Python backend service is left unused in our repository. We found an example[27] of building a RESTful API to manage the DNS resource record based on BIND9 and FastAPI. One suggestion could be to integrate the script generated from the resource record into part of the back-end functionality when a new entity needs to be registered.

- Public access to the DNS server. Currently, we can only test the DNS query when the observer device and the DNS server are running on the same subnet. As pointed out by 2023's team, this could be facilitated by the university providing a dedicated server and domain name to use. However, common security practices like DNSSEC SHOULD be used in this case, the detail of security considerations for the DNS server can be found in draft-reg20 section 7. This part was left unimplemented by us due to a lack of time. We also believe it is important to ask the *DRIP* working group for the best practices in setting up a DNS server.

References

[1] A. Wiethuechter, S. W. Card, and R. Moskowitz, *DRIP Entity Tag (DET) Authentication Formats and Protocols for Broadcast Remote Identification (RID)*, RFC 9575, Jun. 2024. DOI: 10.17487/RFC9575. [Online]. Available: https://www.rfc-editor.org/info/rfc9575.

[2] *The drip det public key infrastructure draft-ietf-drip-dki-03*, draft for drip dki. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-drip-dki/03/.

[3] I. W. DRIP, *Drip-scripts pull request 2*, Accessed: 2024-12-16, 2024. [Online]. Available: https://github.com/ietf-wg-drip/drip-scripts/pull/2.

[4] *Bluetooth 5 extended advertising long range*, Bluetooth Consortium Article regarding extended advertising. [Online]. Available: https://www.bluetooth.com/blog/exploring-bluetooth-5-going-the-distance/.

[5] B. S. I. Group, *Bluetooth 5.4 core specification*. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-amended-5-4/.

[6] *Wifi aware - wifi alliance*. [Online]. Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-aware?utm_source=chatgpt.com.

[7] *Drip entity tags (det) in the domain name system (dns) draft-ietf-drip-registries-20*, drip-registries describes the discovery and management of DRIP Entity Tags (DETs) in DNS. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-drip-registries/20/.

[8] M. Impesi, S. Klasson, M. Larsson, V. Norgren, D. Warlen, and S. M. Yute, "Tdde21 drip 2023," Linköping University, Tech. Rep., 2023.

[9] *Opendroneid observer*, GitHub repo for the ASTM opendroneid observer. [Online]. Available: https://github.com/opendroneid/receiver-android.

[10] *Drip ietf group*, DRIP IETF Group GitHub page. [Online]. Available: https://github.com/ietf-wg-drip.

[11] *Opendroneid github webpage*, opendroneid GitHub web page. [Online]. Available: https://github.com/opendroneid.

[12] N. Semiconductor, *Nrf52840 dk specs*. [Online]. Available: https://www.nordicsemi.com/Products/Development-hardware/nRF52840-DK.

[13] Espressif, *Esp32-s3 specs*. [Online]. Available: https://www.espressif.com/en/products/socs/esp32-s3.

[14] Bitcraze, *Crazyflie spec*. [Online]. Available: https://www.bitcraze.io/products/old-products/crazyflie-2-0/.

[15] A. Wiethuechter, S. W. Card, and R. Moskowitz, "DRIP Entity Tag Authentication Formats & Protocols for Broadcast Remote ID," Internet Engineering Task Force, Internet-Draft draft-ietf-drip-auth-41, Dec. 2023, Work in Progress, 45 pp. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-drip-auth/41/.

[16] B. Wellington, *Secure Domain Name System (DNS) Dynamic Update*, RFC 3007, Nov. 2000. DOI: 10.17487/RFC3007. [Online]. Available: https://www.rfc-editor.org/info/rfc3007.

[17] L. man page, *Linux man page for nsupdate*. [Online]. Available: https://linux.die.net/man/8/nsupdate.

[18] H. Gholami and N. Heydarian Caydan, *Performance measurements on bluetooth 5.2 using nrf52840 development kit*, 2023.

[19] Pixellord, *Support wifi nan esp32-s3*. [Online]. Available: https://esp32.com/viewtopic.php?t=39481.

[20] ascode, *Bluetooth core v5.3 set extended advertising data command*. [Online]. Available: https://stackoverflow.com/questions/74998854/bluetooth-core-v5-3-set-extended-advertising-data-command.

[21] pvvx, *Support for coded phy (bluetooth 5.0 feature) when scanning*. [Online]. Available: https://github.com/hbldh/bleak/issues/1225.

[22] Bitcraze, *Crazyflie repository*. [Online]. Available: https://github.com/bitcraze/crazyflie2-nrf-firmware.

[23] ataffanel, *Disable bluetooth advertising at runtime*. [Online]. Available: https://github.com/bitcraze/crazyflie2-nrf-firmware/issues/17.

[24] N. Semiconductor, *Transmission power*. [Online]. Available: https://docs.nordicsemi.com/bundle/ncs-2.4.1/page/nrf/protocols/matter%20getting_started/transmission_power.html.

[25] *Astm standard*, ASTM Standard. [Online]. Available: https://www.astm.org/f3411-22a.html.

[26] ISC, *Isc documentation for adding new resource record type in bind9*. [Online]. Available: https://kb.isc.org/docs/aa-01140.

[27] *Fastapi example for building dns server restful api backend*. [Online]. Available: https://gitlab.com/jaytuck/bind-rest-api.