TDDE21 DRIP 2023

Mario Impesi   Sebastian Klasson   Martin Larsson
Viktor Norgren   David Wärlén   Samuel Malawi Yute

Contents

1 Introduction

The number of drones in our society is increasing and the capabilities of the drones are becoming more impressive. Civilian drones can be used for legitimate recreational and commercial purposes, but they can also be used in malicious activities, for example spying on military installations or private individuals. Malicious actors need to be held accountable and this creates a need for tying drones to their respective owners. In the same way that there are registration plates on cars that tell authorities who the driver is, there should be an equivalent when it comes to drones. However, having a readable license plate on a drone would be impractical and therefore there needs to be a way of gathering this information from a distance without having direct vision of the drone.

This is where the *Drone Remote ID Protocol* (DRIP) comes in. It allows for the wireless identification and verification of drones and their owners.

This project was done as a part of the course *Advanced Project: Secure Distributed and Embedded Systems (TDDE21)* at Linköping University during the autumn of 2023.

1.1 Project goals

These were the goals that we aimed to achieve in the course:

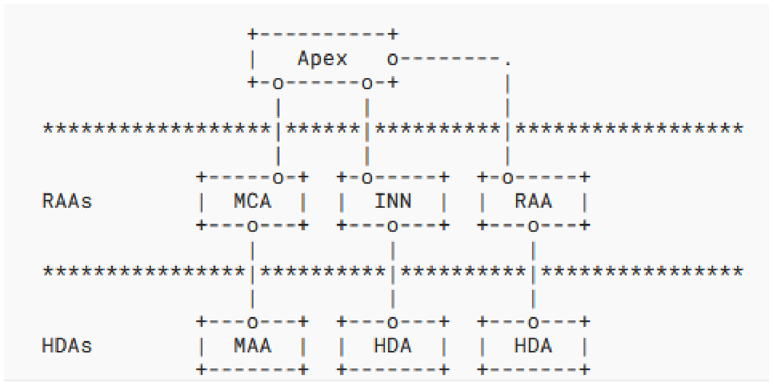
1. Update to RFC9374
2. Port Android App to iPhone (possibly)
3. Interface to DNS registry
4. Authentication extension (update from -17 to -40)
5. Interoperability with other implementations

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 protocol created by IETF with the aim of trustworthy and secure remote identification (RID) of unmanned aircraft systems (UAS) [4]. The idea for DRIP is similar to license plates on cars and driver's licenses: Both UAS and UAS operators have licenses, called unique identifiers, and they are used to uniquely identify UAS and operators, as well as link UAS to operators. The unique identifiers of drones are broadcast using technologies like Bluetooth or Wi-Fi which enable remote actors to receive the identifier and find information about the drone, as well as link the drone to an owner.



```
                 +----------+
                 |   Apex   o--------.
                 +-o------o-+        |
                   |      |          |
*****************|******|**********|*****************
                   |      |          |
               +-----o-+  +-o-----+  +-o-----+
RAAs           |  MCA  |  |  INN  |  |  RAA  |
               +---o---+  +---o---+  +---o---+
                   |          |          |
****************|**********|**********|****************
                   |          |          |
               +---o---+  +---o---+  +---o---+
HDAs           |  MAA  |  |  HDA  |  |  HDA  |
               +-------+  +-------+  +-------+
```

Figure 1: The DIME hierarchy [2]

2.2 DRIP Identity Management Entities (DIMEs)

The structure of the hierarchy is shown in Figure 1. A *Registered Assigning Authority* (RAA) is for example a national civil aviation authority, in the swedish case, Luftfartsverket. *HHIT Domain Authorities* (HDA) are businesses and organizations that provide UAS services. These HDAs register themselves at a RAA and are assigned IDs. The HID is thereby used to tell us which RAA and HDA the entity is registered at. The *Apex* handles assigning the IDs for the RAAs, and acts as the root of the whole hierarchy.

2.3 DRIP Entity Tag (DET)

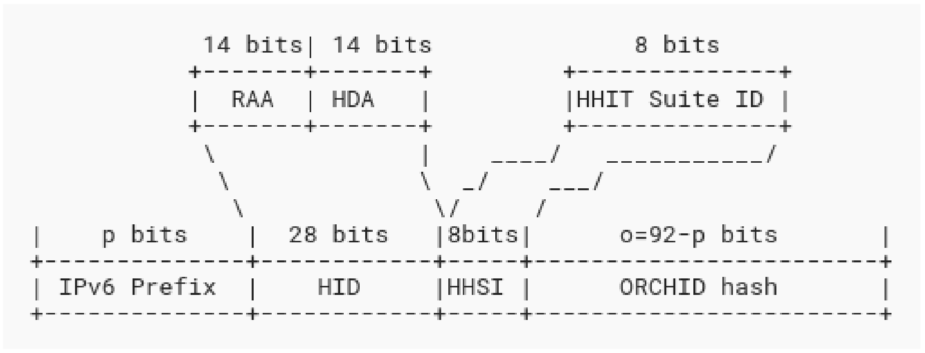
A DET is just a different name for a *Hierarchical Host Identity Tag* (HHIT). The structure of a HHIT is shown in Figure 2.

The prefix is predefined for this type of communication, and has the value: 2001:30/28. The HID contains the IDs of the RAA and HDA that the entity, for example a drone, is registered at.

2.4 DRIP and DNS

One way for people to verify that the drones that they see are registered and safe is by looking them up via DNS. When you receive an endorsement from a drone you can use the included DET to look them up to see if they are registered. Regular people will *not* be able to find any personal information about the drone owner, only if it is registered or not. Law enforcement will have access to more information about the owner, so if it does anything malicious they can find who was responsible for those actions. This can also be used to raise alarms if there are any unregistered drones flying around, thereby you can know in advance which drones to pay extra attention to.

However someone other than the registered owner might have been in control of the drone when it was used maliciously, but this same problem exists when it



```
            14 bits| 14 bits           8 bits
           +-------+-------+        +--------------+
           |  RAA  | HDA   |        |HHIT Suite ID |
           +-------+-------+        +--------------+
            \              |    ____/    ___________/
             \             \  _/    ___/
              \             \/     /
|    p bits    |  28 bits   |8bits|      o=92-p bits       |
+--------------+------------+-----+------------------------+
| IPv6 Prefix  |    HID     |HHSI |      ORCHID hash       |
+--------------+------------+-----+------------------------+
```

Figure 2: Structure of a HHIT [3]

comes to license plates on cars. This system will at least give law enforcement a place to start.

2.4.1 Storing and Lookup

There are two ways to send the DNS messages, either via FQDNs or "reverse DNS lookups as IPv6 addresses per [RFC8005]" [3]. Below we detail the process using the FQDN approach.

Using a DET you are able to construct a Fully Qualified Domain Name (FQDN), which has the structure [2]:

```
{hash}.{oga_id}.{hda}.{raa}.{prefix}.{apex}
```

Here is an example of a FQDN with actual data:

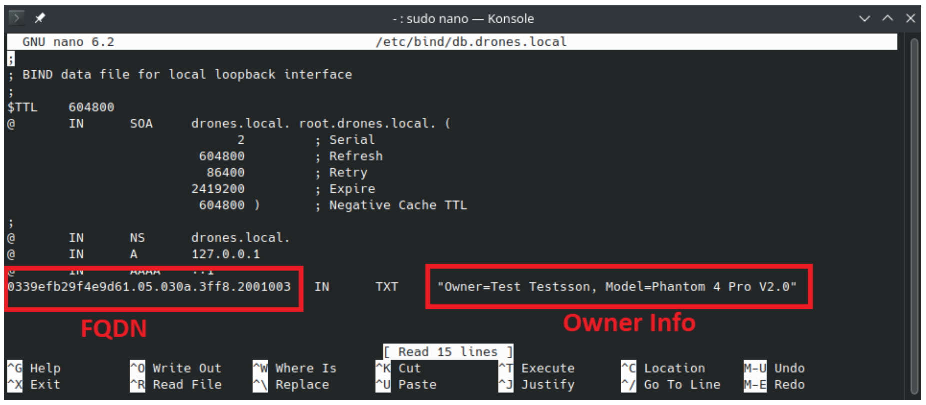
```
c4651542a33fdc26.05.0014.000a.2001003.example.com
```

The FQDN contains all the information necessary to query the DNS server where the drone is registered.

3 Method

We started by just trying to test the previous groups' work to get an idea of how the system functions. We also read a bunch of material that was provided to us to learn more about DRIP and the state-of-the-art.

When we had an idea of how the whole system functioned as a whole, we started working on the tasks we had. We divided the group into pairs and assigned a task to each pair, this allowed each group to focus on one area. We had a group meeting each week so the pairs could update each other on their progress and share relevant information to the other pairs' work.



```
- : sudo nano — Konsole
 GNU nano 6.2                              /etc/bind/db.drones.local
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     drones.local. root.drones.local. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      drones.local.
@       IN      A       127.0.0.1
@       IN      AAAA    ::1
0339efb29f4e9d61.05.030a.3ff8.2001003   IN      TXT     "Owner=Test Testsson, Model=Phantom 4 Pro V2.0"

                                           [ Read 15 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo
```

Figure 3: Illustrates the zone file structure, showing how FQDNs map to TXT records containing drone information.

4 Results

4.1 DRIP and DNS

Our DNS server was configured with a custom zone file. The zone file contained Fully Qualified Domain Names (FQDNs) decoded from DRIP Entity Tags (DETs). Each FQDN was mapped to a TXT record holding the drone information, such as model and owner details, see 3.

To validate and demonstrate the functionality of our DNS server, we utilized the 'dig' command to manually query the DNS for specific drone data. This process involved sending a query for a particular drone's FQDN and receiving the corresponding TXT record in response, see 4.

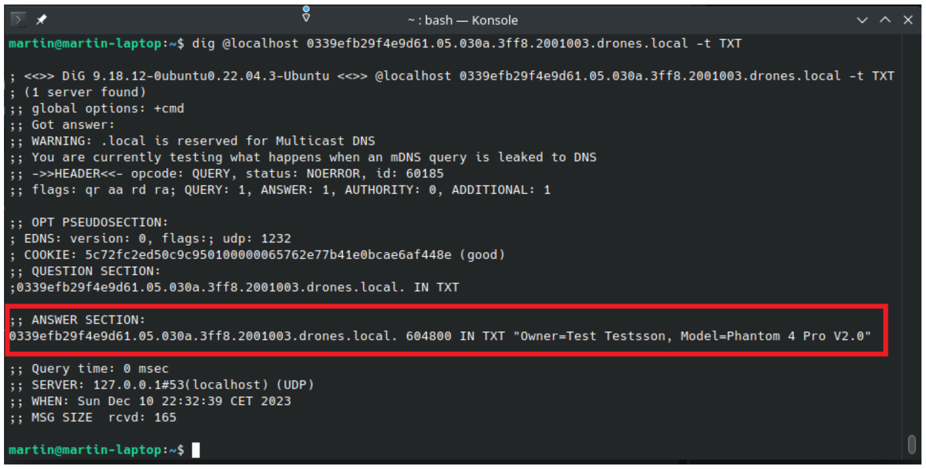
The DNS querying capability was also successfully integrated into the Observer App. This integration allows the app to automatically query our DNS server for information whenever you select a nearby drone. The app retrieves details like the drone's model and owner information and displays it alongside other information, see the app section for more information.

4.2 Authentication

DRIP defines three types of authentication messages, which were implemented as part of this project.

DRIP Link, defined in section 4.2 of [1], was implemented to broadcast the endorsement HDA on UA, so that the observer app could learn the public key of a drone, and use it to read the other two authentication messages: Wrapper and Manifest.

DRIP Wrapper, defined in section 4.3 of [1], was implemented to include the three ASTM messages that the transmitter currently sends: Basic ID type A, Basic ID type D and Location.



```
martin@martin-laptop:~$ dig @localhost 0339efb29f4e9d61.05.030a.3ff8.2001003.drones.local -t TXT

; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> @localhost 0339efb29f4e9d61.05.030a.3ff8.2001003.drones.local -t TXT
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60185
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 5c72fc2ed50c9c950100000065762e77b41e0bcae6af448e (good)
;; QUESTION SECTION:
;0339efb29f4e9d61.05.030a.3ff8.2001003.drones.local. IN TXT

;; ANSWER SECTION:
0339efb29f4e9d61.05.030a.3ff8.2001003.drones.local. 604800 IN TXT "Owner=Test Testsson, Model=Phantom 4 Pro V2.0"

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(localhost) (UDP)
;; WHEN: Sun Dec 10 22:32:39 CET 2023
;; MSG SIZE  rcvd: 165

martin@martin-laptop:~$ 
```

Figure 4: Showcases a sample dig command and its output, effectively retrieving drone information from the DNS server.

DRIP Manifest, defined in section 4.4 of [1], was implemented to include the hashes of the same three ASTM messages included in the Wrapper.

These three messages were sent as ASTM Authentication messages and split in pages to fit the maximum size of a Bluetooth 4 advertisement message.

4.3 App

The observer app builds on the opendroneid implementation which supports all ASTM messages including the four types of messages that our transmitter implements: Basic ID type A, Basic ID type D, Location and Authentication.

From this foundation, the app was successfully extended to implement the DRIP authentication messages as outlined in section 4.2. The app received a paginated authentication message which was then reconstructed.

The messages were also successfully verified with the hashes sent in the DRIP Manifest.

The DNS lookup was implemented by using the FQDN constructed from the DET sent by the UAS.

The current implementation works well with Bluetooth 4 but it was not tested for Bluetooth 5 and WIFI.

5 Discussion

In this section, we reflect on the challenges encountered during our project and the valuable lessons we learned. These insights not only shaped our approach to the project but also provided us with a deeper understanding of the complexities involved in real-world applications of theoretical concepts.

One of the most important aspects of our project was delving into the intricacies of IETF drafts, particularly those concerning drone identification and DNS management. This exploration:

- **Enhanced our understanding** of how protocols and standards are developed and the decision-making behind them.
- **Highlighted the importance** of adhering to global standards for interoperability and future scalability.
- **Challenged us** to interpret and implement these drafts in a practical, working prototype.

5.1 Challenges

Venturing into a relatively new and unexplored domain presented its challenges:

- **Encountering the Unknown:** We often found ourselves in situations where there were no established methods or best practices to follow, pushing us to innovate and experiment.
- **Risk of Misinterpretation:** Without widespread examples to guide us, interpreting and implementing the IETF drafts accurately was challenging.

During this project, we discovered how important it is to break down complex problems into more manageable parts. We did this by using a step-by-step approach which helped us tackle one aspect of a problem at a time and also made it easier to track our progress and identify issues. This also helped us try to reduce the feeling of being overwhelmed by the complexity and scale of the project.

One of the most crucial lessons was understanding the gap between theoretical knowledge and its practical application. Theoretical models often do not account for all the nuances and unpredictable variables encountered in real-world scenarios. We learned the importance of being adaptable and ready to modify our approach when theory did not align perfectly with practice.

5.2 Advice for future students

Create and maintain a document with the full names of all the relevant acronyms. It might even be worth to also add a paragraph with information about the acronym. You are going to be working on this project for a whole semester, there are going to be acronyms that only find two or three times over the whole project, but it can take a lot of time to look them up and find a good summary for them.

Understand that there are two standards in this context: ASTM F3411 and DRIP. DRIP in designed on top of ASTM and DRIP messages are encapsulated in ASTM Authentication messages.

Since this work is cutting edge, there are no FAQs, guides or solutions to find online. A good resource that we used was the IETF DRIP taskforce mailing list (https://mailarchive.ietf.org/arch/browse/tm-rid/) which contains the latest information about the developments and discussions before they update the drafts and RFCs.

6 Future work

- This project implemented three ASTM messages: Basic ID type A, Basic ID type D and Location. The others need to be implemented for a complete message schedule. A suggested schedule is in Appendix B of draft-ietf-drip-auth-41 [1]. An alternative approach could be to use the OpenDroneID implementation[1] of a transmitter and add the DRIP functionality on top of that.
- The transmitter and app were tested using Bluetooth 4. Wifi and Bluetooth 5 need to be tested.
- The draft-ietf-drip-auth-40 has a section on Forward Error Correction. This was not explored or implemented by this group.
- One of the Bluetooth headers, called AD Counter, is currently read by the observer app as a message counter, which counts how many messages of the same type have been sent so far. However at the moment, the transmitter always sends 0 as the AD Counter. This needs to be implemented.
- The observer app should check the previous manifest hash and current manifest hash included in a manifest message, as explained in section 4.4 of draft-ietf-drip-auth-41 [1]. The app can also only receive one type of endorsement in a Link message, which is the endorsement HDA on UA. The app should be able to receive the other types of endorsement explained in section B.2 of [1].
- 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.

[1]https://github.com/opendroneid/transmitter-linux

6.1 DNS

In our project, we successfully implemented a local DNS prototype for drone identification and tracking. There is some potential here for expansion and enhancement. You can follow the README in the drip-backend-dns folder to get started with a local DNS setup for initial testing, and then start building the proper public DNS inside that folder. Note that the code inside ”app.py” and ”run.py” are only placeholder functions carried over from the 2022 work which implemented a blockchain solution to drone management. We have kept the code there to serve as an example of what the DNS functions should do (register, remove, lookup etc). Here are some thoughts on how the development of the DNS can look like:

- **Public DNS implementation:** Our local DNS setup works as a proof-of-concept, but it would be beneficial to implement a public-facing DNS server. This could be facilitated by the university providing a dedicated server and domain name to use.

- **Access Control:** As per the DRIP design, the DNS setup should have access control. The design should allow for varying levels of information to be disclosed based on the user's clearance. For instance, general public queries might return basic information like the drone's model, whereas authorized entities like law enforcement could access detailed data, including the drone owner's name and address.

- **User Interface for DNS Interaction:** Currently, the local DNS setup requires direct interaction with the DNS server. To make this process more user-friendly and efficient, developing a web interface for DNS management is advisable. This interface would allow easy registration, modification, and removal of drone records.

- **DNS record type:** Currently, our prototype uses the TXT record type to store drone information. At the time of writing this report, the IETF DRIP taskforce has suggested to use a part of the unassigned DNS space (TYPE 68) to be used for DRIP and perhaps when you read this it has been decided and you can implement this instead of using TXT records.

References

[1] Adam Wiethuechter, Stuart W. Card, and Robert Moskowitz. *DRIP Entity Tag Authentication Formats & Protocols for Broadcast Remote ID*. Internet-Draft draft-ietf-drip-auth-41. Work in Progress. Internet Engineering Task Force, Dec. 2023. 45 pp. URL: `https://datatracker.ietf.org/doc/draft-ietf-drip-auth/41/`.

[2] Ed. A. Wiethuechter and J. Reid. *DRIP Entity Tag (DET) Identity Management Architecture*. URL: `https://www.ietf.org/archive/id/draft-ietf-drip-registries-13.html` (visited on 12/04/2023).

[3] A. Wiethuechter R. Moskowitz S. Card and A. Gurtov. *DRIP Entity Tag (DET) for Unmanned Aircraft System Remote ID (UAS RID)*. URL: `https://www.rfc-editor.org/rfc/rfc9374.html` (visited on 12/04/2023).

[4] IETF. *Drone Remote ID Protocol (drip)*. n.d. URL: `https://datatracker.ietf.org/wg/drip/about/` (visited on 12/04/2023).