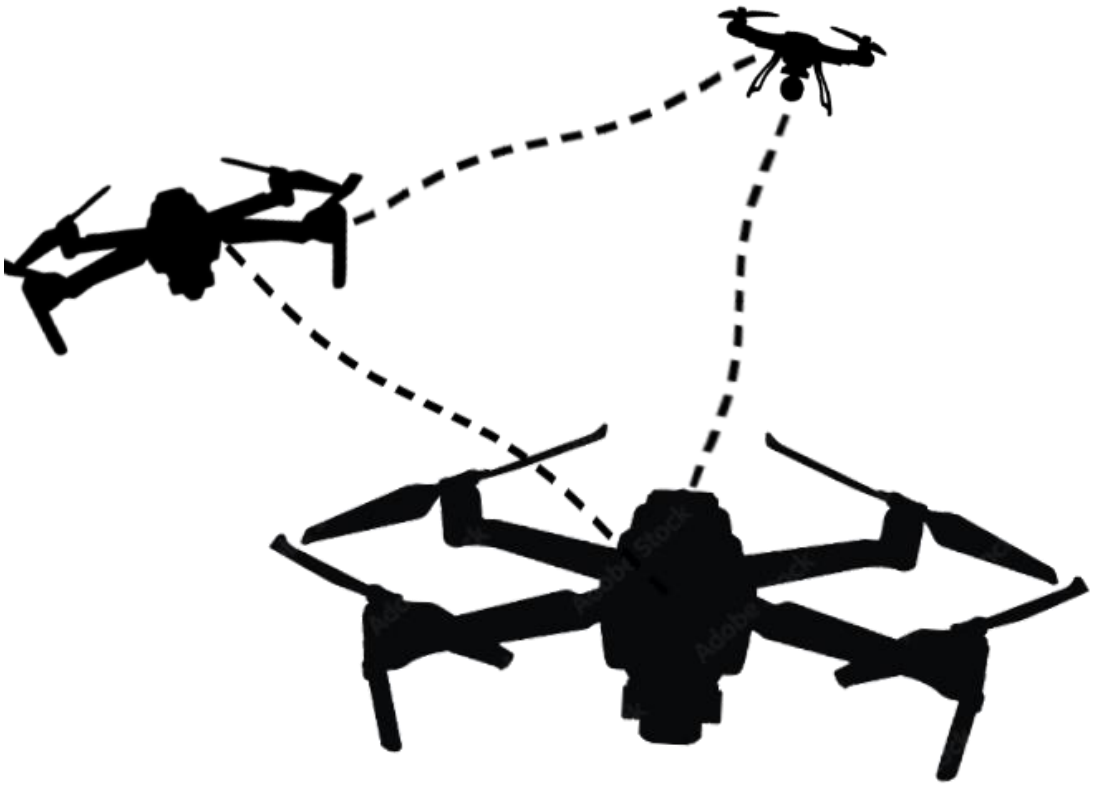
Implementation of Drone Remote Identification Architecture for Unmanned Aircraft Network

Khalid Evans
1907595



Dissertation
Department of Computer Science
Date: 28/04/2023

Declaration

Statement 1

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: Khalid Evans Student (1907595)

Date: 28/04/2023 Student (1907595)

Statement 2

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by citations giving explicit references. A bibliography is appended.

Signed: Khalid Evans Student (1907595)

Date: 28/04/2023 Student (1907595)

Statement 3

The University's ethical procedures have been followed and, where appropriate, ethical approval has been granted.

Signed: Khalid Evans Student (1907595)

Date: 28/04/2023 Student (1907595)

Abstract

As drone adoption is becoming more widespread, it will naturally impose safety and security vulnerabilities as the technology around us continues to evolve. Drones are now being widely used in commercial applications which include supply chains, insurance, construction, and law enforcement. Drones are one of America's most sophisticated weapons; therefore, it is crucial to address the communication security to be improved in the application of Internet of Drones (IoD). This study will investigate the implementation of a security-oriented drone remote identification protocol – specifically called Drone Remote Identification Protocol (DRIP). We will examine and evaluate the effectiveness and reliability of the protocol. We conduct an analysis of the protocol's efficiency and feasibility. We tested the protocol's security using an experimental testbed and a formal verification tool – Scyther. Our results from the formal verification analysis and the real-time testing evaluated the protocol to be in a working condition. Scyther's results proves that the security model is effective against man-in-the-middle attack, replay attack, and impersonation attacks. The results also confirm that both the drone and the observer have successfully exchange their public keys and are using the same flight session key. The implemented protocol meets the requirements for DRIP – from the registries to the private and public lookups. By building a secure foundation for the IoD, we are enabling for a better and safer future of drones that will be used in a variety of different applications: from recreational and commercial to military-based applications.

Acknowledgements

I would like to thank my supervisor for giving me such a learning opportunity to take on a great challenge that spikes my interest in cyber security. Big thanks to my friends and family for encouraging me and supporting me to complete my research at high standards.

Contents

1. Introduction

Drones are an emerging technology that continues to evolve in different use cases: from military to leisure. Its future equally relies on its foundation to evolve and provide a secure infrastructure. The rise in popularity of Unmanned Aircraft Vehicles (UAV) - also known as drones – have resulted in an increase of cyber threats. Today, most drones are integrated with components that utilise cloud, Wi-Fi and/or Bluetooth technology to collect and send data such as video, images, location coordinates, etc. And this concept is called ‘Internet of Drones’ (IoD) (Beekman, 2022).

The increase of drone usage have led to the requirement of regulating Unmanned Aircrafts to address the privacy and security vulnerabilities during their operations. Therefore, the Federal Aviation Administration (FAA) have implemented their first operational rules for drone operators on August 29th, 2016 (Faa.gov, 2022). To address the safety and security concerns specifically, the FAA proposed the integration of Unmanned Aircraft Systems Remote Identification (UAS RID) on December 31st, 2019. UAS RID allows for a drone to provide real-time identification and location information about its flight operation and status to nearby observers and/or authorised parties (Faa.gov, 2019).

The integration of UAS RID enables multiple flight operations to take place when no air traffic services exist. This space is called Unmanned Aircraft System Management (UTM) and is usually set to low altitudes of under 400 feet above ground level (AGL). On the other hand, Air Traffic Management (ATM) that is manned and takes places in altitudes above 400 feet and in airports, military bases, etc.

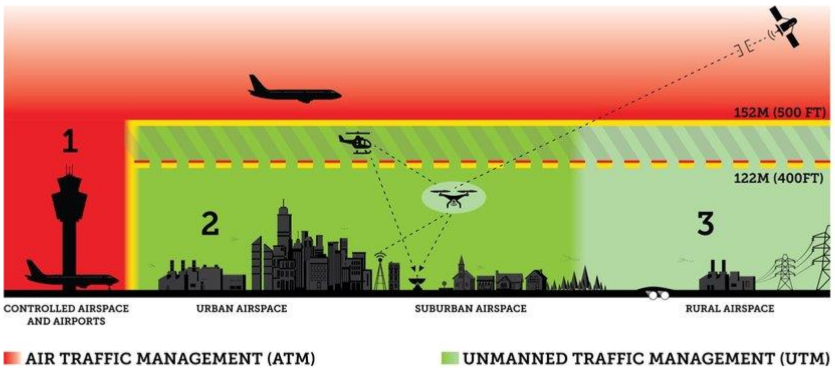

152M (500 FT)
122M (400FT)
1
2
3
CONTROLLED AIRSPACE AND AIRPORTS
URBAN AIRSPACE
SUBURBAN AIRSPACE
RURAL AIRSPACE
🟥 AIR TRAFFIC MANAGEMENT (ATM)
🟩 UNMANNED TRAFFIC MANAGEMENT (UTM)

Figure 1 (ResearchGate, 2018)

In UTM spaces, Unmanned Aircraft System (UAS) operators have real-time constraints to allow them to manage their flight operations safely. The biggest difference between ATM and UTM is that UTM’s primary compliance communication will be between the UAS and the distributed networks through Application Programming Interfaces (API) as opposed to pilots and air traffic controllers communicating via voice (Faa.gov, 2017).

1.1 The Future of Drones

A UAV or drone is defined as an aircraft that is operated remotely without any human pilot on board (Wikipedia Contributors, 2023). Drones were originally developed for military use. But as drone technology evolved using Wi-Fi and mobile network capabilities, alongside improved controls, and interfaces, they expanded to be used in a range of different nonmilitary applications such as outlined by Insider Intelligence (2023). Drones are now being widely used in commercial applications which include:

- **Insurance**: where drone technology can be used to go to hard-to-reach locations when natural disasters happen.
- **Law enforcements**: police can use drones for monitoring expansive open areas. Drones also allow the police to gather insight in dangerous areas without putting officers' lives at risk.
- **Construction**: construction sites need to be surveyed on a regular basis, this process can take 10 hours up to a few days – but with drones, this can be cut down to just a quick 15 minutes.
- **Supply chains**: drones can be used for transportation or delivery of orders to customers in urban areas where there would be congested traffic (Dispatchtrack.com, 2020). Drone can also be used within a warehouse, where it would improve efficiency for inventory management.

According to Jewett (2023), the global use of commercial drones will tenfold by 2030. The drone industry is split into two groups: Military and commercial. With both of those groups are rapidly growing, it comes with new regulation which creates new opportunities for developers and satellite providers for drone technology. The military drone market has a yearly turnover of $12 billion with an expected $17 billion by 2027, which imputes an 8-10% compound annual growth rate (CAGR). On the other hand, the commercial drone market is growing even faster, today's yearly turnover of $8 billion is expected to reach $47 billion by 2028, and that imputes a CAGR of 28 to 30% (Jewett, 2023).

1.2 The Problem Statement

As drone adoption is becoming more widespread, it will naturally impose safety and security vulnerabilities as the technology around us continues to evolve. Drones are one of America's most sophisticated weapons; therefore, it is crucial to address the communication security to be improved in the application of Internet of Drones (IoD).

For example, in 2009, the Iraq insurgents successfully hacked a US drone by using software that was available on the internet for 26$. The insurgents were able to see live video feed of the drone, and this was an issue as they can identify potential targeted zones and evade them (MacAskill, 2009).

In April of 2016, an incident which involves a civilian UAV that allegedly collided with a British Airways plane (Wikipedia Contributors, 2022). The plane was headed to Heathrow airport but lucky, no casualties were reported. This incident shows the importance of geo-fencing airports. By having a registry of authorised drone operations would minimise the risk of these types of incidents.

Another example of an accidental collision was in September of 2017, where a civilian UAV crashed into a helicopter in New York City, United States (Wikipedia Contributors, 2022). The helicopter was able to continue flying till it landed to the nearest airport. Later in December of that year, they investigated the incident to find that the drone operator had flew his UAV away from himself by 2.5 miles where it was beyond visual line of sight (BVLOS). This is one of the reasons that the FAA has prohibited flying drones beyond light of sight.

From the examples noted above, it is clear that the problem with drones is its lack of its regulation. We will address the challenges of real-time response, privacy, authentication, and security. There is a need for a standardised secure drone identification system that identifies and tracks drones in UTM spaces. There are even more risks aside from situational awareness (SA) incidents; the constant risk of drone operators' personally identifiable information (PII) from being stolen or exposed to malicious users, needs a solution. Drones naturally will have a vulnerability of being intercepted with man-in-the-middle attack that tries to spoof signals which causes the drone to malfunction or get hijacked. It is important to understand the different range of potential attacks when designing a solution, which we will expand on in Chapter 2.

1.3 Purpose of Study

This study will investigate the implementation of a security-oriented UAS RID protocol – specifically called Drone Remote Identification Protocol (DRIP). We will examine and evaluate the effectiveness and reliability of the protocol. As stated by Card et al. (2022), with DRIP, the protocol must provide the following:

- Message integrity.
- Non-repudiation.
- Defence against replay attacks.
- Defence against spoofing.

DRIP addresses the security, privacy, and transparency challenges. Privacy and transparency are conflicting goals; therefore, it is important to establish a balanced system.

Card et al. (2022) mentions that DRIP falls under two categories: private and public information:

- Public information which are published in cleartext that would be useful to an observer. An example would be the UAS ID. This information would be publicly available to nearby observers when a drone is in operation nearby.
- Private information that needs to be protected and only accessed by authorised parties. An example would be PII of the operators which contains emergency contacts. This information would be able to be stored in a registry which requires internet connection and authorised access.

To cater to all the issues mentioned above, we will implement a cryptographic mechanism that is resource-efficient and asymmetric. Most existing cryptographic mechanisms are not ideal to be used for drones due to drone constraints of limited processing power, weight, and size. We will implement a mechanism that is designed specifically for IoD network environment to achieve the optimization and security needed.

After implementing DRIP on a prototype, we will finally conduct a performance analysis of the protocol's efficiency, feasibility, and security using a testbed and formal verification tool – Scyther.

1.4 Contribution

This study will address the safety and security issues with drones by implementing a security based UAS RID system. This protocol has the potential to be standardized if proven to be efficient, reliable, and secure. This study will endeavour to explore and discuss this further.

By building a secure foundation for the IoD, we are enabling for a better and safer future of drones that will be used in a variety of different applications: from recreational and commercial to military-based applications.

We are contributing to the development of secure drone UAS RID architectures by implementing a solution of our own - using DRIP as our protocol foundation. It is equally important to compare and acknowledge other solutions which we will cover in Chapter 3.

1.5 Definition of terms

This section will cover all the terms that will be using throughout this document. This may be used as a reference for DRIP terminology.

Term	Definition
UAV	Commonly referred to as ‘drone’, is an unmanned aerial vehicle that does not have an on-board pilot that can fly autonomously or remotely (Wikipedia Contributors, 2023).
UAS	Unmanned aircraft system is made up of a UA with the required components such as launch and recovery equipment and C2 links (Card et al., 2022).
UA	Unmanned aircraft -also known as ‘drone’- is an aircraft with no pilots that operates remotely or autonomously (Card et al., 2022).
VLOS	Visual line of sight is the straight-line view that is not obscured by buildings, trees, or other natural obstacles (www.DroneTraining.co.nz, 2020).
BVLOS	Beyond visual line of sight is when the drone is outside visual range distance of the operator pilot (Unmanned Systems Technology, 2022).
SA	Situational awareness means knowing ‘what’s going on’ in the context of your drone’s operation in an environment (Skybrary.aero, 2014).
ATM	Air traffic management is the management of air traffic which includes three main services: air traffic services (ATS), air traffic flow management (ATFM), and airspace management (ASM) (Skybrary.aero, 2021).
UTM	Unmanned Aircraft System Management is the traffic management of uncontrolled UAS operations at low altitudes (Card et al., 2022).
AGL	Above ground level means the height difference from the ground over which a drone is flying (Drone Pilot Ground School, 2019).
Observer	An entity – which is not necessarily an individual – that is within VLOS of a drone (Card et al., 2022).
Operator	A natural person that is operating or planning to operate a drone (Caa.co.uk, 2023).
IoD	Internet of drones is described as a network architecture that provides control and access between drones and users (Abdelmaboud, 2021).

GCS	Ground control system that is part of the UAS. The GCS is what the pilot uses to remotely perform C2 with the drone (Card et al., 2022).
GPS	Global positioning system is a satellite-based radionavigation that gives positioning, navigation, and timing services (Gps.gov, 2023).
API	Application programming interface is a set of definitions that enables communication between two software components (Amazon Web Services, Inc., 2023).
PII	Personally identifiable information is referred to as the personal data of the operator or the observer in a UAS RID context (Card et al., 2022).
Location vector	Provides the drone's location, direction, and altitude. Location vector could also include the speed and status of the drone in operation (Card et al., 2022).
UAS ID	UAS identifier is the unique identity to the UA (Card et al., 2022).
UAS RID	UAS remote identification and tracking is a system that enables observers to detect drones that are in operation (Card et al., 2022).
Operation	A flight plan that could include a series of flights under the same UAS (Card et al., 2022).
4-D	Means the four-dimensional data: longitude, latitude, altitude, and time (Card et al., 2022).
DAA	Detect and avoid, also known as sense and avoid (SAA) is a safety system that is integrated in drones to avoid collision against buildings, powerlines, etc (Unmanned Systems Technology, 2022b).
C2	Command and control, in a UAS context, it refers to the communication between the GCS and the drone (Card et al., 2022).
GNSS	Global navigation satellite system transmits satellite-based positioning and timing data to receivers (Europa.eu, 2016).

2. Technical Background

2.1 Attack Vectors

As mentioned, drones naturally impose a security and safety vulnerability that can be exploited by malicious users or attackers. The impact of drone attacks on the public and in high integrity applications -such as commercial and military use- can impede the drone adoption in those applications. There is a need for a security oriented UAS RID protocol and DRIP is a solution that we will explore and experiment in this document. First, we need to identify the various types of drone attacks that could be carried out which can be divided into two categories: network layer attacks and physical layer attacks.

Starting with network layer attacks:

- **Flooding attack**: also known as Denial of Services (DoS) attack, consumes a lot of system resources by sending high volumes of traffic to make the drone's network service unavailable to receive data packets from genuine users (Wang et al., 2021). This attack can be performed by sending too many ping commands to the drone that it starts to use all its resources to reply back, hindering the drone's main functionality. Because of drones limited resources, they are specifically more vulnerable to flooding attacks. The result of a flooding attack can render a drone failing to receive GCS commands which will paralyze it, leading it to possibly crash.

- **De-authentication attack:** This attack occurs between a drone/user and a Wi-Fi access point. Most operators perform the C2 link between the drone and the GCS through Wi-Fi, and this attack takes place specifically on the 802.11 protocol which is not entirely secure (Wang et al., 2021). Attackers can send a spoofed de-authentication frame to disconnect it from the drone. This can easily be done by sniffing for the address of the drone through wireless network sniffing. After sending the spoofed de-authentication packet to the GCS, it will disconnect the drone from the operator; therefore, allowing the attacker to take control over the drone.

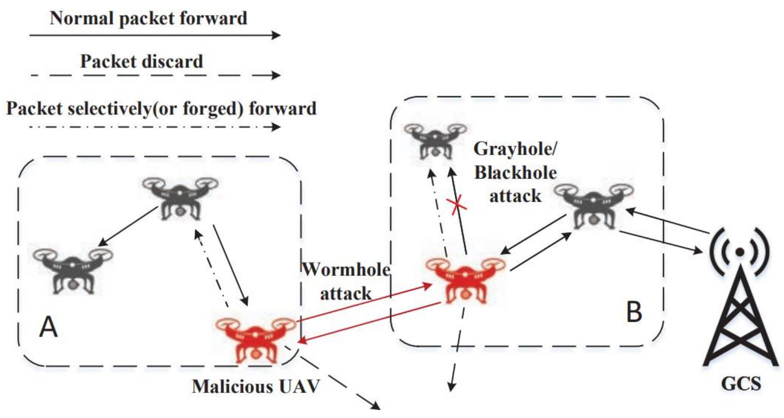


Figure 2 - Routing attack (Wang et al., 2021)

- **Routing attack:** This attack occurs mainly in a swarm of drones in a given network. This takes places when a malicious or unauthenticated drone adds itself to the network acting as an authenticated drone. The routing attack then starts by making the malicious drones act as optimal nodes. This allows the attacker to influence the routing protocol as the other genuine drones will choose the malicious drones to relay their packets through them. This attack is more damaging when a swarm of drones are present in a network. There are three types of routing attacks: black hole, gray hole, and wormhole attacks.

- **Black hole attack:** When an unauthenticated malicious drone joins a network of drones and drops all the packets from going through the other drones, a black hole attack occurs. The packets dropped are supposed to hop through one drone to another to send the command(s) to all drones, but this attack will affect it with a partial or complete loss of data transmission. This can be damaging as drone can be led to crash due to collision or total energy consumption (Wang et al., 2021). Another vulnerability is the possible exposure of confidential information which would be contained in the data packets being sent.

- **Gray hole attack:** This attack takes place when an unauthenticated malicious drone selectively chooses to drop or forward specific packets as opposed to blocking off all the packets. This type of attack is much harder to detect compared to a black hole attack. Another case is where a malicious drone forwards forged packets to the genuine drones to identify more potential attack vectors.

- **Wormhole attack:** This attack takes place when a pair or multiple unauthenticated malicious drones have made their way in a network to exploit the genuine drones. Exploitation occurs because with multiple malicious drones, the ability to act as optimal nodes for communication is greater in this case allowing for them to be the first to receive the data packets. When multiple or a pair of malicious drones communicate with each other, they can selectively forward specified packets to the rest of the drones (Wang et al., 2021). A wormhole attack also has the ability to receive packets from both the source and the recipient addresses.

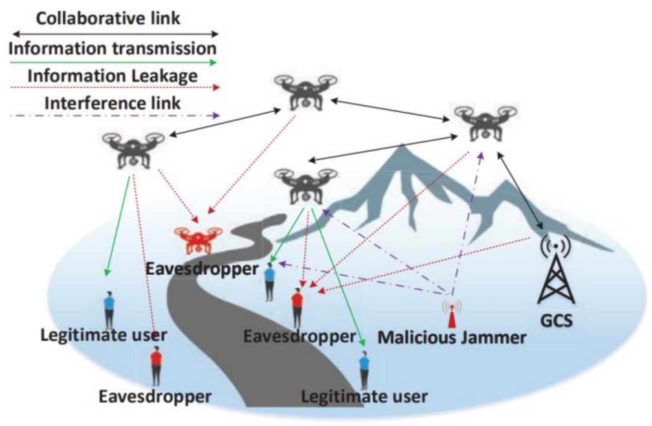


Figure 3 – physical layer attacks (Wang et al., 2021)

Followed are the physical layer attacks which is known as the physical link attack.

- **Passive eavesdropping attack:** This attack takes places when a malicious drone or observer awaits in a wireless channel to sniff and collect any data – that could include PII's. Passive eavesdropping is a silent attack that has no effect on the rest of the genuine entities in a network (drones and observers), which makes it difficult to detect when it is present (Wang et al., 2021). This attack is a form of a reconnaissance attack.
- **Active eavesdropping attack:** On the other hand, active eavesdropping is less passive and silent by including an offensive presence which could include signal interference and jamming. As you can see in figure 3, a malicious entity can jam the communication between genuine drones and the GCS by sending interference signals. The affect of this attack can be detrimental on functionality of the drones which could impede or even paralyse drones from executing commands (Wang et al., 2021).

2.2 Drone Constraints

A UAS compromises of a mobile device with embedded sensors and subcomponents; therefore, they are subject to limitations to their processing power and their resources. Drones' physical and technological constraints include:

- **Range**: The maximum range a drone sustains a communication signal. This can be affected by obstacles, terrain, buildings, etc.
- **Battery life**: This depends on the drone size; the drone's battery dictates the battery consumption rate and the duration a drone can execute a flight operation.

- **Payload capacity**: Refers to the maximum weight a drone can carry. This includes the different types of cameras, sensors, motors, and processing components.
- **Processing power**: This is the computational capacity to perform complex calculations in a duration of time. This affects the drone's performance to DAA obstacles. Processing power depends on the drone's CPU, GPU, and RAM specifications.

Not any just cryptographic mechanism can be used to secure the drone-to-operator communications. Traditional security protocols that are commonly used to authenticate Internet of Things (IoT) devices does not apply to be equally efficient in the context of IoD. An example of a widely used protocol is digital certificate-based authentication mechanism. X.509 is the most commonly used type of certificates that are designed for servers and computers which do not have resource constraints (Cho et al., 2020). Drones are heavily reliant on the functionality of its sensors and receiving correct the real-time data from the GCS. Therefore, we must implement a secure and equally resource efficient solution for drone communications that would not greatly affect its core functions.

2.3 Network RID and Broadcast RID

As stated by Card et al. (2022), there are two means of UAS RID:

- **Broadcast ID**: defines a set of messages for the UA/drone to transmit locally through Bluetooth or Wi-Fi to be received to nearby observers in real time.
- **Network ID**: Required constant internet connection where all the information of the UAS is accessed globally online through servers to be queries by observers.

Network ID relies on constant internet connection, and this requires a Network RID Service Provider (SP) and Display Provider (DP) as servers that are well-connected infrastructure. As these requirements are not fully adopted yet. We will focus on Broadcast ID which can work without internet access, as the communication link is a one-way direct transmission from the UA to the observer as you see in *figure 4*.

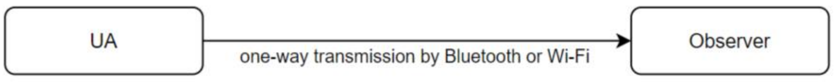


Figure 4 – Broadcast ID

Internet connectivity with Broadcast ID is only involved when the observer wishes to use the transmitted UAS ID as the primary key to further look up information in the registry server, in our case, it would be through a mobile application which communicates to the server via Application Programming Interfaces (API). Whereas in Network ID, it requires three steps as you can see in figure 5:

- UA/GCS to SP
- SP to DP
- DP to observer

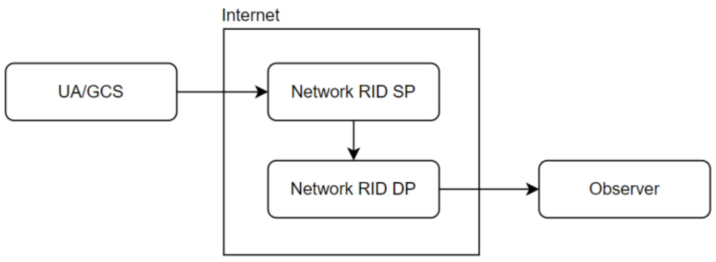


Figure 5 – Network ID

2.4 The Proposed Protocol: DRIP

As mentioned previously, drones are vulnerable to numerous attack vectors that needs to be addressed in an RID system to be adopted as a new standard for drones. The main concern is the safety and security of drones, operators, and observers. A security-oriented protocol must achieve the following (Card et al., 2022):

- Safeguard the operator's privacy.
- Enable effective authentication.
- Allow real-time access to information for authorised bodies.

The above is achieved with DRIP as it addresses the security and safety challenges including allowing information obtained from UAS RID to be immediately actionable.

DRIP has requirements that needs to be met that we will cover in detail in Chapter 4. The focus of DRIP's protocol is the following:

- **Provable ownership requirement**: This address that the sender is who they claim they are. This is achieved in this document using asymmetric public key infrastructure (PKI) which will be covered in Chapter 7. This requirement protects against spoofers who try to imitate an authorised entity.
- **Encrypted transport requirement:** End-to-end encryption is a requirement during the transport of private data. This covers the communication links between drones, observers, and registries.
- **Registries:** Registries is a fundamental to DRIP. A registry is a trusted server that saves the extended information that is not being continuously transmitted between drones and observers. It is the management of identifiers which is organised by the UAS ID's. The UAS ID is a unique identifier that will be used as the unique key to lookup extended information about a drone/operator.
    - **Public lookup:** DRIP enables lookups that is not restricted by the party submitting the query to access the public information of a given UAS ID. We will cover more about the public information data stored in Chapter 4.
    - **Private lookup:** Likewise, DRIP enables lookups that are restricted to only be accessed by authorised bodies to access private information from the UAS ID. More on this in Chapter 4.

3. Related Work

It is important to mention the existing research as this will help us identify gaps and potential contradictions and build a theoretical framework of our proposed protocol. We will carry out an overview and a critical analysis on some relevant studies on UAS RID protocols that are based on security and authentication.

Starting with a research paper by Lei et al. (2021) - a study that proposed an optimized lightweight identity security authentication protocol called Optimised Identity Authentication Protocol (ODIAP). The protocol focuses on tackling the security challenges faced in IoD networks. ODIAP compromises of 3 phases and 7 authentication processes, which uses public keys and session keys. The protocol anticipates for both remote and physical attacks and it treats all communication networks as untrusted by default to combat impersonation attacks, replay attacks, etc. The proposed protocol is forward and backward secure, designed to not compromise the system if one message was cracked. Performance and real-time communication are also a requirement in the study; therefore, the protocol implemented is designed to have a computation that is completed in a limited time. The frequency and size of the data required is kept at a minimum to not affect its streamlined communication and efficiency. The research also has carried out a formal verification using the industry-renowned verification tool, ProVerif. They have proved using ProVerif that an attacker cannot compromise the system, satisfying the forward and backward security requirement.

Looking at another research by Kashif Naseer Qureshi et al. (2022), which proposes a message authentication protocol. This protocol is called Efficient Authentication Scheme for Safety Applications for Internet of Vehicles (EASSAIV). The research focuses on both Internet of Vehicles (IoV) and UAV's and addresses their security and privacy concerns with their implemented protocol – EASSAIV. The protocol uses asymmetric-key encryption to secure communication between UAV's and vehicles. The protocol compromises for 3 phases: authentication phase, key exchange phase, and the data transfer phase. The protocol uses timestamps and the nonce value that is appended along the message in the implementation phase that is to be sent to the controller. This is used to prevent attacks such as replay attacks. The protocol is designed so that the vehicle's virtual identity (private key and ID) is generated by the controller and sent back to the vehicle once the owner accesses the vehicle. The protocol focuses on message integrity by ensuring the messages received are not altered during communication. This is achieved by using the hashed value of the shared session key. Data integrity is then verified by the receiver by calculating the hash using the same algorithm used by the sender. The study also conducts an informal security analysis and is validated using a security tool called AVISAP to show that the proposed protocol is safe and working condition. EASSAIV concludes with strengths against replay attacks, password guessing attacks, multiprotocol attacks, and reflection attacks.

Another study by Cho et al. (2020), which focuses on the security and safety of UAV operations. The study proposes a protocol - that tackles threats caused by unauthorised UAVs – called SENTINEL (Secure and Efficient Authentication for Unmanned Aerial Vehicles). The protocol generates flight session key per drone operation that is stored in a centralised database. The flight session key is used as the message authentication code (MAC). Instead of using the conventional X.509 v3 certificate, SENTINEL uses lightweight certificates by containing minimum information required to generate a public key pair. The SENTINEL certificate size was evaluated by a size of 186 bytes whilst an X.509 certificate is 682 bytes. The research also designed the certificates to be used in its binary format as apposed its human-readable text format, thus resulting in even more size reduction and a reduced memory footprint. SENTINEL uses ECDSA (Elliptic Curve Digital Signature Algorithm) with SHA-256 for its digital signature algorithm. The study used ProVerif for their formal security analysis. The results proved confirms that no attacker was able to obtain the shared flight session key. The study also proves that both parties are using the same generated flight session key. An informal security analysis was conducted, and it was evaluated that its security features include mutual authentication, session key agreement, replay attack, man-in-the-middle attack, impersonation attack, known session key attack, and more.

A study by Ko et al. (2021), proposes two sub-protocols for UAV-to-UAV and UAV-to-GCS. These protocols are mainly designed for military environments. The protocol to solve the gaps in UAV secure communication and provide support for forward secrecy, and prevention of repudiation attacks. The study mentions the importance of tackling the security concerns with unauthorised access and malicious control. The two proposed sub-protocols are called SP-D2GS (Security Protocol for Drone-to-Ground Control Station) and SP-D2MD (Security Protocol for Drone-to-Monitoring Drone). In this study, the GCS is where the private and public keys are generated, followed by a certificate request (CSR) to the certificate authority (CA). The cryptographic mechanisms of the protocols include using ECDSA for the digital signature algorithm's function, hash-based message authentication code (HMAC), and pseudo-random number generators (PRNG). The study listed the requirements of their protocol include protection against denial of service (DoS) attacks, protection against man-in-the-middle (MITM) attacks, non-repudiation, confidentiality, integrity, strong key exchange, and mutual authentication. The protocol uses timestamps alongside the digital signature to authenticate that the message is indeed from the GCS. Then, the receiver also calculates the HMAC value to compare with the received hash value to verify that the message integrity is kept. The study has conducted two formal verifications using BAN-Logic and Scyther which concluded that both SP-D2MD and SP-D2GCS are proven to satisfy the security requirements mentioned above – secure key exchange, integrity, and data authentication of messages.

Another study by Hashem, Elmedin Zildzić and Andrei Gurtov (2021), which proposes a permission-based blockchain framework as a replacement of registries that DRIP requires. The framework - which is called Hyperledger Iroha - uses smart contracts, permissions, and multi-signature transactions. The proposed blockchain framework consists of multiple blocks that uses signed transactions by the private key to verify the sender. Each block has a timestamp. By using a blockchain framework, the protocol will naturally inherit its characteristics which are the following: anonymity, decentralisation, persistency, and auditability. All the nodes in the blockchain are aware of each other and the active accounts. A copy of all the nodes' public keys is stored in each node. Hence, the framework uses asymmetric-key encryption to sign and verify transactions. The study does mention a privacy concern in a scenario where a node is compromised. The study then talks about the preventative measures to reduce the consequences which include using an encrypted hard drive whilst keeping the encryption key in another offline hard drive. The study mentions that with more active nodes, the faster the transaction time will be, likewise, with more active UAS's, the more transactions will occur hence increasing the response time. When a transaction is initiated, it will randomly select a node to distribute the load on the blockchain. The study concludes with a performance analysis which concludes with the fact that the blockchain framework was successful in meeting the DRIP requirements: public lookup, private lookup, provisioning (sending of static/dynamic of drones), AAA policy. The study concludes with the weakness of this framework, as there is a requirement of continuous real-time location updates, this will impose a big requirement of storage especially in bigger blockchain networks.

Another study by Ahokas and Persson (2022), that proposes a protocol that implements DRIP by focusing on session key secrecy and message authenticity. The proposed protocol uses HIT (Host Identity Tag) which is part of HIP (Host Identity Protocol) because of its features that ensures message integrity and authenticity. The study also mentions that extremely low probability of a collision to occur. The protocol also uses EdDSA (Edwards-curve Digital Signature Algorithm) to generate the keys in quickly and securely. The protocol is designed to meet the DRIP requirements, the study further talks about the protocol security: it must receive secret messages and reject false messages, and all messages received must be verified that it came from the claimed sender without any message tampering. The study then models the proposed DRIP protocol in Tamarin – a formal verification tool – by establishing the rules and schema. They have split this process into two sections: establishing a secure connection with a HIP Base Exchange, and implementation of DRIP features such as broadcasting the RID. The study concludes with the Tamarin examination results that attacks such as man-in-the-middle attacks and impersonation attacks through a drone's leaked private key is not possible. The study mentions the gaps in their formal verification and talks about lack of some of DRIP's functionalities in the Tamarin model which would provide a complete verification of the DRIP process. Some of the functionality that lack in the Tamarin model include first-time drone registration, certificate functionality (using EdDSA), registries functionality. This study does prove that the proposes version of DRIP is secure and in working condition.

4. Requirements

DRIP gives confidence for Identification Friend or Foe (IFF). As mentioned by Card et al. (2022), protocol that we will implement must cover the following objectives:

- Preserve the UAS operator's privacy.
- Enable a secure dynamic connection between the observer and the UAS.
- Enable immediate use of actionable information by authorised parties.

4.1 DRIP Requirements

Registries are an essential requirement for DRIP. DRIP defines its data in two private and public data. The continuous communication between the drone(s) and observer(s) must not transmit any private information. The private information is to be stored in the registry itself. In our case, registries will be servers that can be accessed to the public (unauthorised people) through APIs to perform public lookups, and to authorised bodies (police, government, etc.) which can perform private lookups. Below is an example of two tables queried by a specific UAS ID. The table in figure 6 is the result of a public look up, and the table in figure 7 is the result of a private look up. It is very important to store sensitive PII's on the registries where they are safer and far less likely to be in the hands of malicious users where they can use it to spoof the information in the signal message headers when they are not the claimed sender. The registry addresses the provable ownership requirement for DRIP. The registry ID is used to identify which registry an entity is registered in. Likewise, the entity ID is to identify which entity instance it is in a given registry.

UAS ID	**Serial No.**	**Registry ID**	**Entity ID**
9877DRONE112	AB12345678	3334A	0001

Figure 6 – public lookup table

UAS ID	**Operator Name**	**Emergency Contact**	**Date of Birth**
9877DRONE112	John Smith	07888888100	12/06/1995

Figure 7 – private lookup table (the information above is imaginary as an example)

As you see above, public, and private lookups give different information depending on the authority level.

4.2 Specification

To complete our aims for this project, we must devise a formal list of requirements to meet the acceptance criteria of our project. The following are the three main points that enables us to implement DRIP in a working system with an observer entity, the drone entity, and the registry lookup server.

- A drone Broadcast RID module (raspberry pi) that securely communicates to the ground station or observer on an end-to-end encrypted transport - in this project, the ground station or observer is a mobile application.
    - The module will transmit static information which will include the UAS ID, and dynamic information which will include the location vector, timestamp, angle, and ground speed.
        - The static information will need to be transmitted once at least each 3 seconds.
        - The dynamic information will need to be transmitted once each second and no older than a second.
- A registry server for public and private look up which is only accessed by authorised users to retrieve personally identifiable information (PII)
- The ground station - which is the mobile application which will be the user interface that will act as a radar to show all drone entities in the general area and access to the registry server which requires internet connection.

As mentioned above, DRIP specifies transmitting static and dynamic information. Below, as illustrated by figure 8 are the lists of what each type of information must include:

Static Information
UAS ID that is generated by the registry.
UA Serial Number generated from the manufacturer.
Registry ID: to identify which registry the entity is in.
Dynamic Information
Location Vector of the UA (longitude, latitude, and altitude)
Ground speed in mph
Angle of the UA clockwise from the North
Timestamp including date and time
UA emergency status

Figure 8 – static and dynamic information

The registry server must include a database that can be queried for public and private look ups. This accordingly requires two tables that use the UAS ID as the primary key.

UAS ID	**Serial No.**	**Registry ID**	**Entity ID**

Figure 9 – public lookup schema

UAS ID	**Operator Name**	**Emergency Contact**	**Date of Birth**

Figure 10 – private lookup schema

During the development of DRIP, there has been au update to the specification which includes two new tables to be added to the registry as you can see by figures 11 and 12:

hashed_ID

Figure 11 – registered entities schema

ID	**public_key**	**hashed_ID**	**address**	**entity_type**

Figure 12 – certificate authority schema

The two new tables were needed to store the information of the UAS once it registers to the registry. Here is the information that is stored from a registered UAS in the figure 12:

- ID: A numeric identifier that will be used to keep entries in order in the database.
- Public_key: The received RSA public key generated by the UAS.
- Hashed_ID: The unique identifier that will be used by the UAS to authenticate itself during the authentication phase – more on this in Chapter 7.
- Address: The latest updated IP address of the UAS drone.
- Entity_type: This can either be ‘drone’ or ‘observer’.

5. Methodology

As this project required software engineering and testing, it is important to plan the Software Development Life Cycle (SDLC) of this project. SDLC provides a structed process that enables us to produce the software needed to project requirements in the most efficient amount of time possible. The SDLC that we chose for this project is Agile SDLC methodology.

5.1 Software Development Life Cycle (SDLC)

Agile methodology breaks the projects several phases to make it manageable. It works of the basis that the project will be constantly worked on throughout the development life cycle (Wrike, 2022). Agile is one the most software development methodologies for project management due to is adaptability to change.

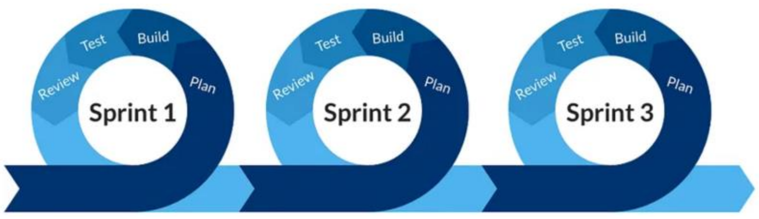


Figure 13 – Agile development (Agrawal, 2019)

Agile methodology has been used throughout the development of this project. This chosen SDLC combines both iterative and incremental methods. We have used the iterative methodology to build a little of each part of the protocol at a time. And using the incremental methodology to deliver completed components and features throughout the project duration. The advantages of using Agile methodology for the project (Wrike, 2022):

- Continuous cycles: Using this principle to break down tasks into smaller iterations, and continuously testing and refinement throughout each cycle of development.
- Flexible evolution: This principle allowed the project to adapt to specification requirement changes. As the project development progressed, the original specification has been amended which resulted in a change of priorities which we will cover in the next section of this chapter.

5.2 Work Schedule

Below, is how the project was originally broken down into simpler tasks. As we are using the Agile SDLC, we wanted to list out the projects into sub components and further break down those components into tasks. During the project development, each sprint had a duration of 2 weeks, one week of development followed by a week of testing. Below is the initial table of tasks and with the estimated time to complete them, followed by an illustration of the initial Gantt chart.

Task Number	Task	Days to complete
Task 1	Design the drone entity system in Raspberry Pi	5
Task 2	Implement communication over Wi-Fi	10
Task 3	Create broadcasting of static and dynamic information	10
Task 4	Design Registry database	5
Task 5	Create Registry server and create tables for private and public lookups	5
Task 6	Create API for registry	10
Task 7	Design mobile application	5
Task 8	Implement mobile application to receive broadcasted information from drone entity	10
Task 9	Create radar interface for observer(s) to view nearby drones	5
Task 10	Create registry public look up interface by using UAS ID as primary key on mobile app	5
Task 11	Implement registry private look up interface by adding credentials for authorised parties	5
Task 12	Implement Certificate Authority in registry server that creates lightweight certificate	10
Task 13	Implement Flight sessions key system to be used as a MAC	5
Task 14	Implement MAC and lightweight certificate cryptography on the Raspberry Pi drone entity	15
Task 15	Implement MAC and lightweight certificate authentication in the mobile app	15

Figure 14 – Table of tasks

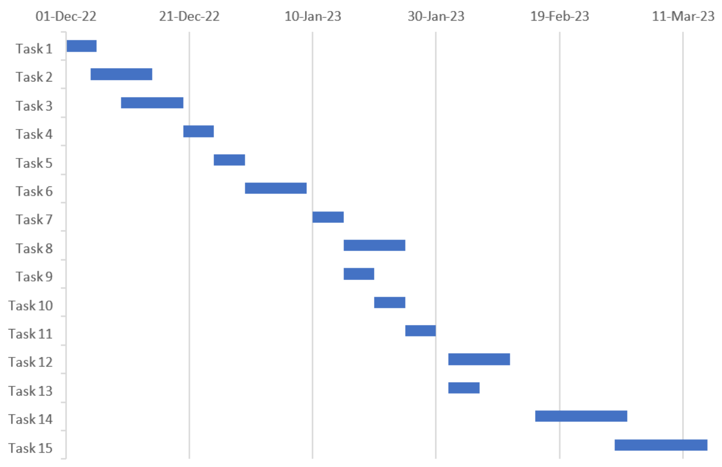


Figure 15 – Gantt chart

As you can see by figure 14 and figure 15, the project was planned out for a rough estimate of the development duration. Reflecting back on this, after finishing the implementation of the project – which we will cover in detail in Chapter 7 – the project took longer than the estimated ranges. The specification had gone through some amendments which resulted in a change in priority of tasks – more on this in Chapter 4. The task table (figure 14) was very helpful as a start to understand the scope of the project, but as we progressed, we strayed away from the original chronological order planned. The project was faced with technical obstacles which costed more time than anticipated. We took a different approach to the project which was a more incremental methodology. The DRIP implementation project requires 3 components(entities) to be developed:

- Drone entity (Raspberry Pi)
- Observer (mobile application)
- Registry (server)

With an incremental process, we focused on delivering a component fully before moving to the next. Incremental testing allowed for changes to be made to the code as it was being developed, which resulted with reduced overall risk of the project.

5.3 Deliverables

Throughout the project schedule, as mentioned in the initial document, it was split into sprints of 2 weeks which required serval deliverables per sprint:

1. New version release of the code
2. Document that summarises work done, and tasks finished.
3. Updated specification (if amendments have been made)

As the project progressed, we produced the required deliverables per sprint which helped with back tracking code and keeping track of tasks completed in project timeline. You can refer to those deliverables in the Appendix.

5.4 Quality Assurance Planning

During the development of the project, quality assurance was important to help identify issues early on and improve the code quality to be optimised, organised, and understandable. Here are the measures that were taken to ensure the quality of the project implementation:

- Test functions: These functions were used as unit tests and debug functions that is used to pre-populate tables in the database to ensure it accepts the right type of information and declines incorrect data submission. Here is an example of a debug function below as illustrated by figure 16.

```
#Populate the certificate authority database with a dummy entry
def debug_populate_registered_entities():
    with open("./key.pem", "rb") as file:
        k = file.read()
    key = Fernet(k)
    key.decrypt(get_registered_entities()).decode()
    count = len(['result'])
    ID = count + 1

    connection = sqlite3.connect('registry.db')
    cursor = connection.cursor()
    cursor.execute("""
    INSERT INTO certificate_authority
    (ID, public_key, hashed_ID ,address ,entity_type)
    VALUES ({},'test_public_key','test_hashhh','test','drone')
    """.format(ID))
    connection.commit()
    cursor.close()
```
Figure 16 – debug function to populate table.

- Code commenting: It is important for the code to be efficient to meet the specification, but it is equally important to make it understandable to others. This makes the code maintainable long term. Comments are left on top of each method to describe what it does with some mid-method comments to explain the more complex algorithms.

6. Preliminary Studies

DRIP is a security oriented UAS RID protocol. To successfully implement DRIP, there are 2 main requirements that must be covered cryptographically in the context of secure communication:

- Message confidentiality
- Message integrity

To tack the message confidentiality, we have used symmetric encryption for the communication between the server and the drone, and between the server and the observer. We have implemented asymmetric encryption using RSA public-private key pairs that is applied on the communication between the drone and the observer.

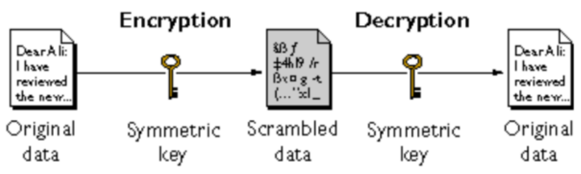


Figure 17 – symmetric-key encryption (Ibm.com, 2014)

Symmetric-key encryption is where the same key is used for both encryption and decryption. We have used this encryption between the server and entity (drone or observer) communication. This type of encryption is used in this case because we know the two parties are trusted. But there is still a drawback to symmetric keys: exchanging the secret. But we solved this by pre-embedding the entities with the 128-bit AES (Advanced Encryption Standard) key 'from factory', and this removes the dependency of relying on another key to exchange the key. Symmetric-key encryptions are less complex to process, and the keys are small but provides enough level of protection for our application (Ibm.com, 2014). The key is 128-bit AES because it is faster and more secure than its predecessor – DES (Data Encryption Standard). AES is also being used by United State's National Security Agency (NSA) for their top-secret information confidentiality (Wikipedia Contributors, 2023a).

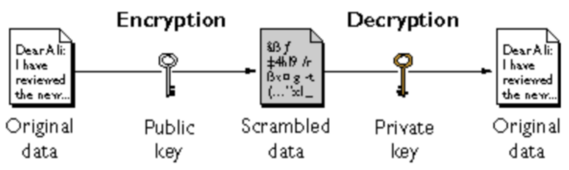


Figure 18 – public-key encryption (Ibm.com, 2021a)

Asymmetric-key encryption involves using a private and public key pair that is generated per party – this concept is called public-key cryptography. The key pair is based on algorithms by Rivest-Shamir-Adelman (RSA). The public key is generated to be published and used to encrypt messages, whilst the private key is kept secret and is used to sign or decrypt messages. Signing a message using the private key's digital signature, and this ensures that the encrypted message can also include the sender's authentication. The importance of using public-key encryption in a setting is the requirement of message confidentiality so that it would not be understood or be altered by a middleman. The RSA public key pairs we used are 2048-bit (Ibm.com, 2021a). Generating an RSA public key pair is based on two prime numbers and an auxiliary value. The security of the RSA depends on the 'factoring problem' which is the difficulty of figuring out the product of those two prime numbers. Till today, there is no known method used to crack the system if a large key is used, hence the reason of using 2048-bit-sized key pair (Wikipedia Contributors, 2023b).

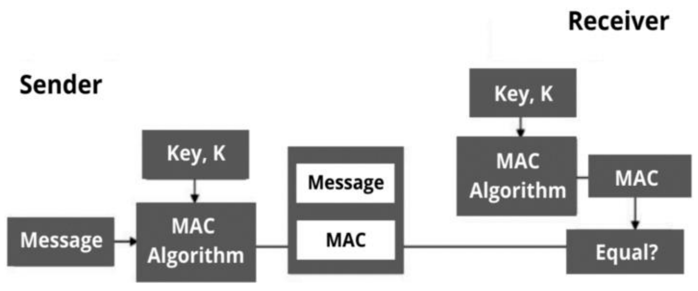


Figure 19 – Message authentication code (Usemynotes, 2022)

Message authentication code (MAC) is a tag that is sent along the original message to the receiver. MAC are used to ensure message integrity during the transmission of data. MAC is also referred to as checksum. It uses a shared symmetric key between the sender and the receiver to calculate and authenticate the message. We have used the hash-based message authentication protocol (HMAC) using the SHA-256-bit hash function. HMAC is similar to digital signatures but it instead of relying on the public key pair (asymmetric keys), it only uses a shared symmetric key *(Usemynotes, 2022)*.

Using the HMAC, the original message is hashed into a message digest (MD) using the SHA-256 algorithm, then it will be encrypted using a symmetric key *(Usemynotes, 2022)*. In our case, we call it *flight session key* – more on that in Chapter 7. The hashed message is then appended along the original message to be sent to the receiver. The receiver will then use the original message to recalculate the hashed message and will then check whether the received MAC is equal to the one it calculated. This would prove that the message has not been altered since it has been sent from the sender.

7. Implementation

7.1 Network Model

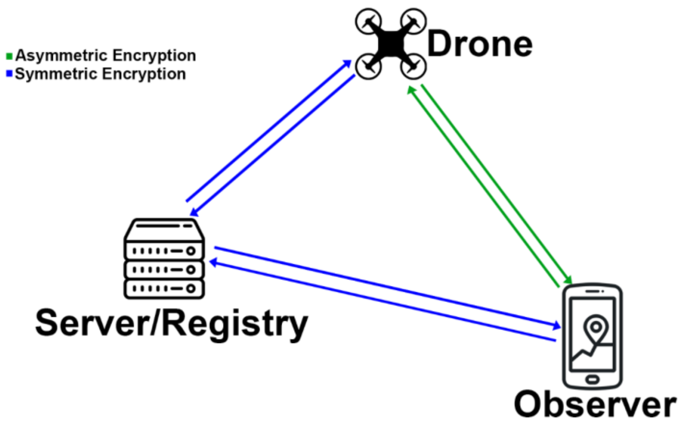


Figure 20 – Network Model

As you can see from Figure 20, the network model uses both types of encryptions: symmetric and asymmetric. At the initial stage of the protocol, the drone and observer entities communicate with the server with the symmetric key encryption which is embedded in their systems 'from factory'. The server is the middleman that shares the public keys of each other alongside the generated flight session key. The design of the system is set up where the public key pair is generated by the entity (drone or observer) then it sends its public key when registering for the first time. After registering, the server generates a hash that it stores and sends a copy to the drone/observer that will then be used as a login credential to 'go live'. Once both entities have exchanged public keys, the communication between the drone and the observer is done in asymmetric encryption and using the flight session key as the MAC. This ensures message confidentiality and integrity.

7.2 Threat Model

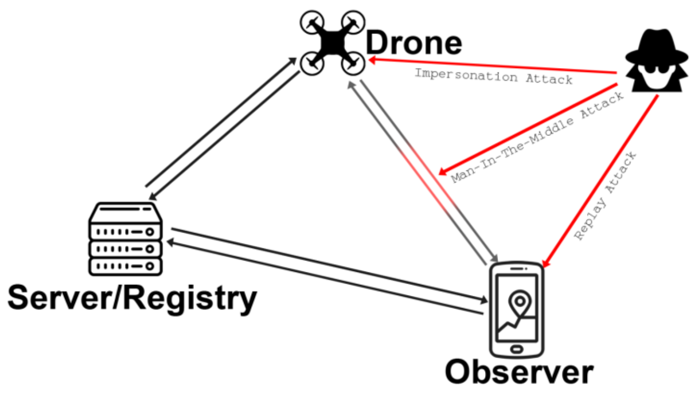


Figure 21 – Threat Model

From the threat model, we can anticipate the different types of attack vectors that are possible during an operation. As illustrated by Figure 21, there are three highlighted threat vectors:

1. Impersonation attack: Where an attacker can either act as a genuine drone to relay information from an authorised drone or even completely impersonate a genuine drone to receive messages that potentially carry PII's.
2. Man-In-The-Middle attack: An attacker can be silent and listen to the data as it passes by. This is particularly hard to detect as the attacker is being passive.
3. Replay attack: This occurs when an attacker wants to deceive an observer by either sending old or altered message packets. This can trick an observer that a certain drone is authorised or in some other location when it is not.

7.3 Security goals

The protocol proposed is designed to meet DRIP's requirements which are confidentiality, integrity, availability, and non-repudiation. With every security measure, there are trade-offs. In our proposed protocol, it is the cost of time over confidentiality. The protocol ensures the message's privacy and the message's integrity to not be altered during communication. The goals for the protocol to cover the attack vectors that we mentioned in Chapter 2. By using public key pairs, flight session keys, and timestamps; the protocol's security should meet the requirements of DRIP with certainty. It is important to test security design theory with a formal verification tool to simulate attacks and expose any gaps in the cryptographic mechanisms, more about this in Chapter 8.

7.4 Cryptographic Mechanisms

```
def generate_hashed_ID(mac_address, serial):
    message  = '{}|{}|{}'.format(mac_address,serial,generate_two_digits()).encode()
    print(message)
    hashed_ID = hashlib.sha256(message).hexdigest()
    return hashed_ID
```

Figure 22 – generate hashed ID function

```
def generate_flight_session_key():
    digits = string.ascii_uppercase + string.digits
    session_key = ''
    for _ in range(32):
        session_key += secrets.choice(digits)
    return session_key
```

Figure 23 – generate flight session key function

```
#Creates the hashed message using a message authentication code (MAC)
def hash(session_key, message):
    session_key = session_key.encode()
    message = message.encode()
    hashed_message = hmac.new(session_key, message, hashlib.sha256)
    return hashed_message.hexdigest()
```

Figure 24 – HMAC function

```
#Encrypts the plain text into cipher text using the public key into utf-8
def encrpyt(message):
    with open('public.pem', 'rb') as p_file:
        public_key = rsa.PublicKey.load_pkcs1(p_file.read())
    encryptped_message = rsa.encrypt(message.encode(), public_key)
    return encryptped_message
```

Figure 25 – RSA encryption function

Implementing the cryptographic mechanisms were mostly done in Python as you can see from the code snippets above. Starting with Figure 22, generating a hashed ID which will be used the login credentials for an observer/drone, is created using the MAC (media access control address) address with its serial number and two random digits that the server generates. The hashed function is then sent back to the observer/drone, but the entity will never know what the two random digits were so that even when an entity gets physically compromised, it wouldn't be able to replicate and generate more genuine hashed ID's.

Figure 23 shows the generation of a flight session key which is 32 in length and made up from letters and numbers. Only one flight session key can be used between a drone an observer. And that session key only lasts during the ‘live’ operation. This flight session key acts as the shared secret between the drone and the observer.

Figures 24 and 25 illustrate the hash-based MAC function and the RSA encryption function. The HMAC uses SHA-256 algorithm as is it more securely than its predecessors. The RSA public key pairs are of 2048-bit size to also ensure key secrecy and complexity.

7.5 Registration and Authentication Process

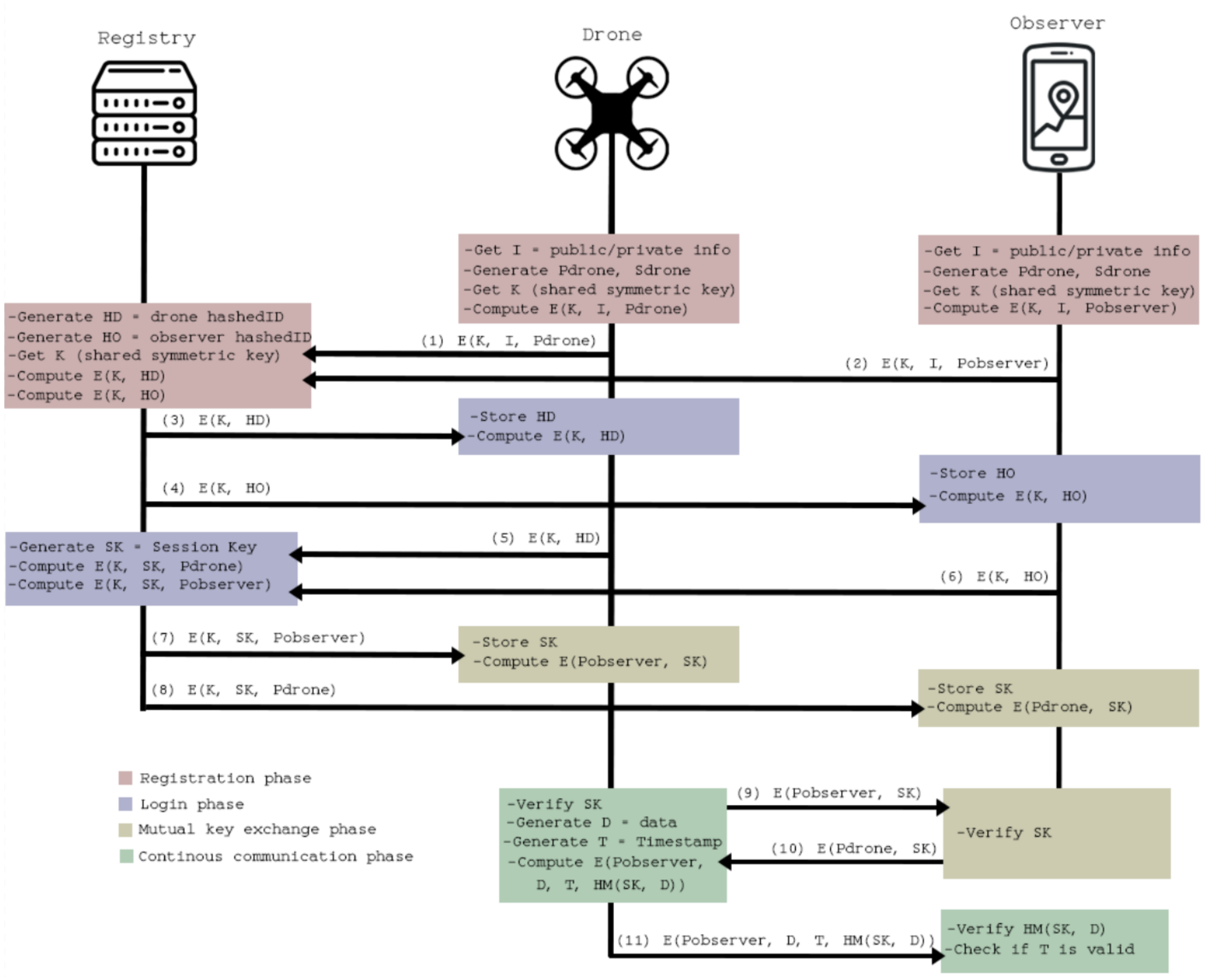


Figure 26 – Protocol phases

As Figure 26 illustrates, the multiple protocol phases:

- Registration phase: Both the observer and the drone register themselves into the server by sending their information which includes their public keys.
- Login phase: The server will then generate hashed ID and send them to the observer and drone. Both the observer and drone have to resend their hashed ID back to the server to successfully ‘log in’ and be added to the ‘live’ entities array.
- Mutual key exchange phase: Once the server detects a pair of observer and drone entities that are live, it will generate a session key and will send it to each entity along with the public key of their pair. So, the observer will get the public key of the drone and vice versa. Finally, the drone and observer will send each other the session key

using each other’s public keys. If the session key is confirmed equal, then they have both verified their connection.

- Continuous phase: This is where the drone continuously generates the dynamic data and bundles it with the static data. The drone will send the original message along a hashed version of the message using the session key as the symmetric key of the hash function. The drone will encrypt the whole message with the observer’s public key. Once the observer receives the message, it will verify the hashed message by comparing it to its own calculation of hashing the original message. It will then verify the timestamp is not and expired or repeated or invalid date to protect against replay attacks. If it passes both checks, the observer is sure the message is kept untampered and is verified from the drone. Then the observer will wait for the next continuous message in a loop.

7.6 Continuous Communication Process

```
#Send the continous data using UDP socket
def send_message(address, message):
    port = 8008
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.sendto(str(message).encode(), (address, port))

#Send a message at a rate of 10 ticks per second.
def continous_messaging():
    while True:
        message = get_data()
        print("Sending '{}' to '{}'".format(message, OBSERVER_ADDRESS))
        send_message(OBSERVER_ADDRESS,message)
        time.sleep(0.1)
    return
```

Figure 27 – UDP message sending function

As you can see above, the two functions to send a UDP (User Diagram Protocol) packet to the target address. The tick rate was set to run at 10 ticks per second. DRIP’s requirements mention that the continuous communication should be at a frequency of a minimum of 1 per second for 80% of the time (Card et al., 2022). A message sent is comprised of two sections: dynamic and static information. The static information that data that will – and should – not change during an operation, and this includes the UAS ID, serial number, and registry ID. The dynamic information updates as frequent as the sensors update their readings. The dynamic information includes the 3-D vector (longitude, latitude, and altitude), ground speed, angle, timestamp, and status of the drone. The purpose of a drone’s status is to give extra context to the observer to improve their situational awareness. Examples of status messages include: “In operation”, “Inactive”, “Ending operation”, “Patrolling”, “Responding to emergency”. The way the protocol is designed ensures for scalability as it can accommodate a change in message formats to add more information or change their order. We achieved this because the messages are sent in JSON format – which is an open standard for data storage that contain attributes and arrays (Wikipedia Contributors, 2023b).

7.7 (Physical) Anti-Tampering Process

```
while True:
    # get modified time
    current_modified_time = os.path.getmtime(protected_file_path)
    time.sleep(5)

    # get modified time again
    updated_modified_time = os.path.getmtime(protected_file_path)

    # Check if protected file has been modified
    if updated_modified_time != current_modified_time:
        # Trigger watch dog - Enter self destruct actions here!
        print('Unauthorised tampering detected - Self destruct activated')
```

Figure 28 – Anti-tampering function

As you can see, Figure 28 illustrates the physical anti-tampering 'watch dog' that is designed to constantly scan the targeted file directory for any modification. This part of the implementation is out of scope of DRIP, but it still is important to mention the equal importance of physical security. Once triggered, the system is only set to print a message alerting the operator/user about the event. Other behaviours considered was self-destructing the entity so that when in hands of a malicious user, the entity wipes all the data – which includes the symmetric keys, the encryption algorithms used, the server address, etc.

8. Analysis

8.1 Experimental setup

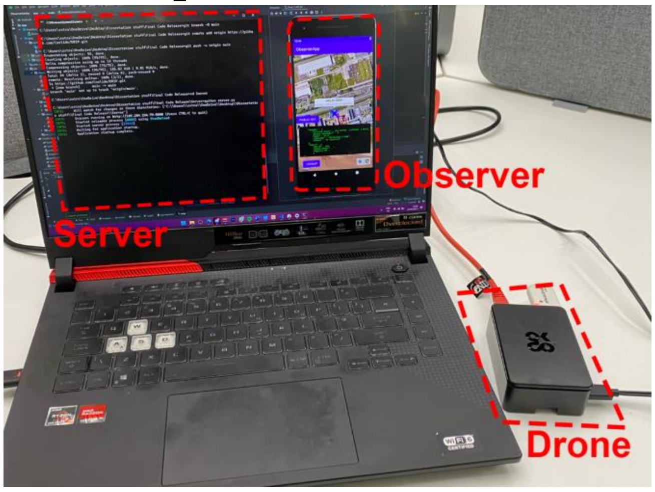


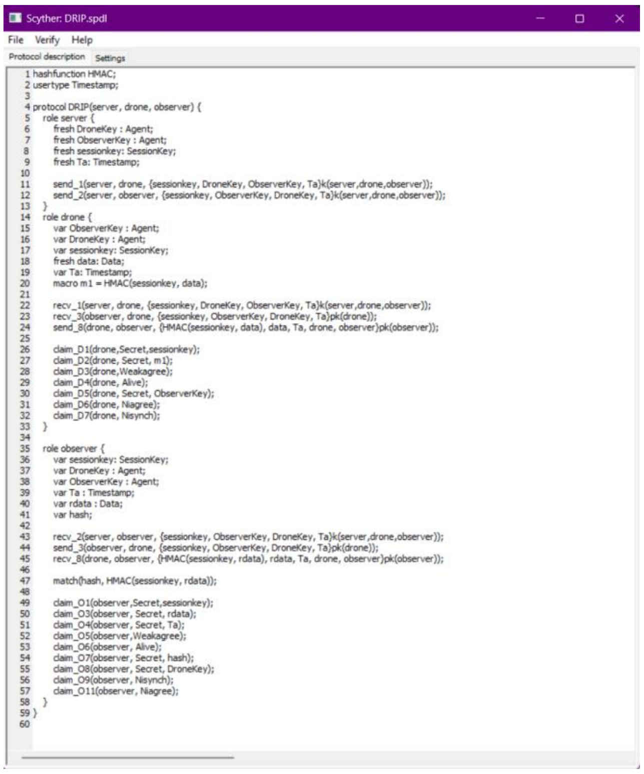
Figure 29 – Project setup

As mentioned, this project covers three entities: drone, observer, and a server. Here are detailed specifications for each platform used to run each entity.

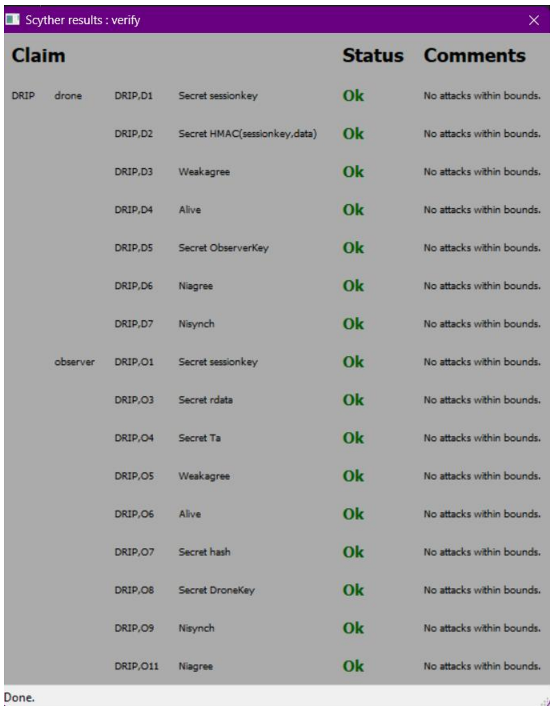
- Drone: Raspberry Pi 4 Model B Rev 1.4, 1.89GB RAM, Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC 1.8GHz | Implemented using Python 3.9.2.
- Server: Microsoft Windows 11 Home | 10.0.22621 N/A Build 22621 | 16GB RAM, AMD Radeon RX 6800M, AMD Radeon RX 6800M | Interpreted using Python 3.11.0 and web server implemented by Uvicorn and FastAPI.
- Observer: Android Studio Pixel 2, Version11.0 (R) API 30 (emulated) | 1.5GB RAM, 4 cores | Implemented using Kotlin.

Due to not having a physical Android device, the project faced difficulties with implementing some of the cryptographic mechanisms to working condition, hence the project resorted towards using simulated data to complete a running prototype of the application. As both the emulated Android device and the server/API are hosted on the same machine, it was difficult to assign different addresses to each entity.

8.2 Formal Verification


Scyther: DRIP.spdl

```
hashfunction HMAC;
usertype Timestamp;

protocol DRIP(server, drone, observer) {
  role server {
    fresh DroneKey : Agent;
    fresh ObserverKey : Agent;
    fresh sessionkey: SessionKey;
    fresh Ta: Timestamp;

    send_1(server, drone, {sessionkey, DroneKey, ObserverKey, Ta}k(server,drone,observer));
    send_2(server, observer, {sessionkey, ObserverKey, DroneKey, Ta}k(server,drone,observer));
  }
  role drone {
    var ObserverKey : Agent;
    var DroneKey : Agent;
    var sessionkey: SessionKey;
    fresh data: Data;
    var Ta: Timestamp;
    macro m1 = HMAC(sessionkey, data);

    recv_1(server, drone, {sessionkey, DroneKey, ObserverKey, Ta}k(server,drone,observer));
    recv_3(observer, drone, {sessionkey, ObserverKey, DroneKey, Ta}pk(drone));
    send_8(drone, observer, {HMAC(sessionkey, data), data, Ta, drone, observer}pk(observer));

    claim_D1(drone,Secret,sessionkey);
    claim_D2(drone, Secret, m1);
    claim_D3(drone,Weakagree);
    claim_D4(drone, Alive);
    claim_D5(drone, Secret, ObserverKey);
    claim_D6(drone, Niagree);
    claim_D7(drone, Nisynch);
  }

  role observer {
    var sessionkey: SessionKey;
    var DroneKey : Agent;
    var ObserverKey : Agent;
    var Ta : Timestamp;
    var rdata : Data;
    var hash;

    recv_2(server, observer, {sessionkey, ObserverKey, DroneKey, Ta}k(server,drone,observer));
    send_3(observer, drone, {sessionkey, ObserverKey, DroneKey, Ta}pk(drone));
    recv_8(drone, observer, {HMAC(sessionkey, rdata), rdata, Ta, drone, observer}pk(observer));

    match(hash, HMAC(sessionkey, rdata));

    claim_O1(observer,Secret,sessionkey);
    claim_O3(observer, Secret, rdata);
    claim_O4(observer, Secret, Ta);
    claim_O5(observer,Weakagree);
    claim_O6(observer, Alive);
    claim_O7(observer, Secret, hash);
    claim_O8(observer, Secret, DroneKey);
    claim_O9(observer, Nisynch);
    claim_O11(observer, Niagree);
  }
}
```


Scyther results : verify

Claim				Status	Comments
DRIP	drone	DRIP,D1	Secret sessionkey	Ok	No attacks within bounds.
		DRIP,D2	Secret HMAC(sessionkey,data)	Ok	No attacks within bounds.
		DRIP,D3	Weakagree	Ok	No attacks within bounds.
		DRIP,D4	Alive	Ok	No attacks within bounds.
		DRIP,D5	Secret ObserverKey	Ok	No attacks within bounds.
		DRIP,D6	Niagree	Ok	No attacks within bounds.
		DRIP,D7	Nisynch	Ok	No attacks within bounds.
	observer	DRIP,O1	Secret sessionkey	Ok	No attacks within bounds.
		DRIP,O3	Secret rdata	Ok	No attacks within bounds.
		DRIP,O4	Secret Ta	Ok	No attacks within bounds.
		DRIP,O5	Weakagree	Ok	No attacks within bounds.
		DRIP,O6	Alive	Ok	No attacks within bounds.
		DRIP,O7	Secret hash	Ok	No attacks within bounds.
		DRIP,O8	Secret DroneKey	Ok	No attacks within bounds.
		DRIP,O9	Nisynch	Ok	No attacks within bounds.
		DRIP,O11	Niagree	Ok	No attacks within bounds.

Done.

Figure 30 – Scyther script

Figure 31 – Verification results

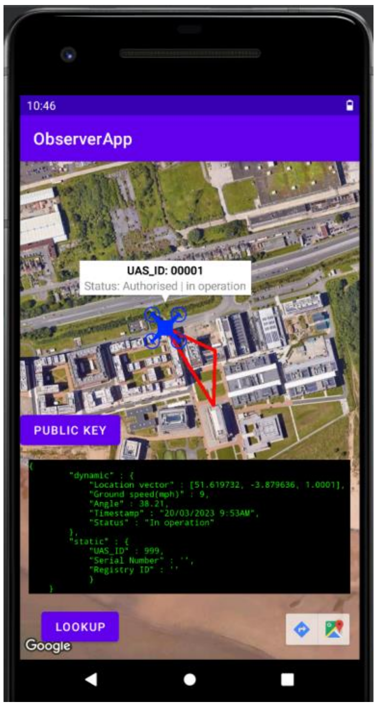
Scyther is well-respected automated formal security verification tool. The nature of the project being about cyber security deems this an essential step to evaluate the proposed protocol. Scyther is used to test and verify protocols and analyse its behaviours through simulation (Cremers, 2014). As you can see from Figure 30, the script written to model the project's proposed protocol. The script implemented the hash function, the public key pair, the symmetric key – that is presented by *k(server, drone, observer)* which denotes that it is a shared secret between all three entities. The sequence of events starts from the server sending to the drone and observer the public key of each other followed by the flight session key. Next, the drone receives a confirmation message from the observer as it contains the session key that is encrypted using the drone’s public key. After the confirmation step, Scyther simulates a message - that contains *data* – to be sent to the observer. The message *data* is hashed using the session key to generate the hash that will be appended to the message. Finally, the message also contains a timestamp and is encrypted using the observer’s public key.

To ensure the protocol model is secure from attacks and does not have any vulnerable gaps: claims are written under the drone and observer to test whether a security attribute holds.

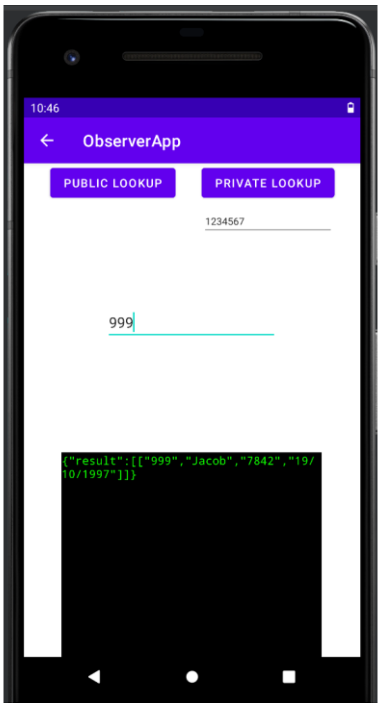
- Secret claim: It is a statement that holds if the sender believes the message is kept secret after sending it over to the receiver. Scyther checks if the property holds by checking attacker traces from analysing it.
- Alive claim: This assertion holds if the entity has been proven not idling during the protocol execution. This claim is used to ensure that each entity is active during the communication.
- Weakagree claim: This claim is particularly useful with ensure that all the entities have the same shared values such as the session key. Weakagree claim ensures that all the roles still have the same shared values by the end of protocol execution but is not a strong assertation as to whether the values came from the correct sender.
- Nisynch claim: It is an assertion that claims the role is synchronised with the other roles during the protocol execution. This is another important claim to prove whether the protocol is aligned or not. The model specifically includes the timestamp for this reason, but this claim is a not a strong assertion as to whether the received messages are fresh. They could still be performing replay attacks but have passed this claim due to being sent in a synchronised way. This is why it is important to use multiple claim assertions to thoroughly analyse a protocol.
- Niagree claim: This claim is similar to Weakagree as it checks that all the other roles and itself have the same shared value and are kept consistent after the protocol execution.

As Figure 31 illustrates, the verification analysis resulted with not attacks within bounds and have passed all the claims. This proves that the proposed protocol does meet the requirements for DRIP in terms of secrecy and integrity of the messages sent between the observer and the drone. The analysis evaluates that the protocol is in working condition and that it is not possible for any of the mentioned attacks to occur.

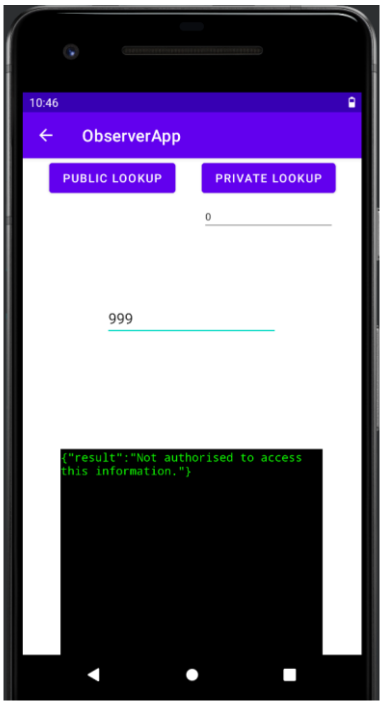
8.3 Experimental testbed



```
{
        "dynamic" : {
            "Location vector" : [51.619732, -3.879636, 1.0001],
            "Ground speed(mph)" : 9,
            "Angle" : 38.21,
            "Timestamp" : "20/03/2023 9:53AM",
            "Status" : "In operation"
        },
        "static" : {
            "UAS_ID" : 999,
            "Serial Number" : '',
            "Registry ID" : ''
            }
    }
```



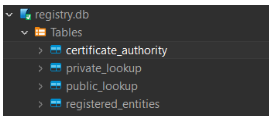
```
{"result":[["999","Jacob","7842","19/10/1997"]]}
```

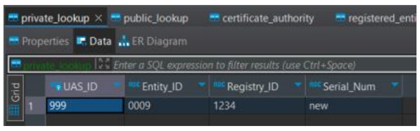


```
{"result":"Not authorised to access this information."}
```

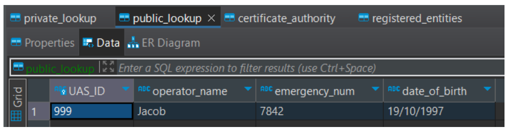
Figure 32 – App sceenshot1 *Figure 33 – App sceenshot2* *Figure 34 – App sceenshot3*

The project's experimental testbed was faced with some difficulties in implementing all the features and specifications originally planned. This was due to the issues mentioned before alongside the difficulty of implementing correctly as we are using different programming languages to communicate together. To ease the development of the prototype, we have decided to simulate some of the aspects of it to meet the time constraints of creating a prototype. Nonetheless, as you can see from the figures 32, 33, and 34 are screenshots from the mobile application. Figure 33 is the main activity once you launch the app, it uses Google Maps to plot the drone on a terrain map view and it also creates line traces so that the user can see which path it as taken so far. Towards the bottom of the screen, you can see a black box with JSON data which is the simulated data used for testing the mobile application. Looking at Figure 33, is the lookup activity, where the observer can make GET requests to the API of the server to retrieve either public or private information about an operator – using their UAS ID as the primary key of the query. To query a private lookup, you are required to be authorised, but due to this being out of scope of the project, the API only checks if it's a predetermined number or not. Further ideas that would be secure to implement for authorised users – is a one-time-password (OTP) system that is generated to the authorised user as a singular use token to query a private lookup. As you can see in Figure 34, the observer has received a negative response from the server due to the invalid authentication token. All the information is stored in an SQLite 3 database as illustrated in Figures 35, 36, and 37.





	UAS_ID	Entity_ID	Registry_ID	Serial_Num
1	999	0009	1234	new



	UAS_ID	operator_name	emergency_num	date_of_birth
1	999	Jacob	7842	19/10/1997

Figure 35 *Figure 36* *Figure 37*

9. Conclusion

The Internet of Drones is only growing and is evolving in more use cases from military to commercial and to industrial environments. This project was aimed to contribute to the research of Unmanned Aircraft Systems Remote Identification (UAS RID) and the exploration and experimentation of Drone Remote Identification Protocol (DRIP). The project has covered all the requirements of DRIP and was evaluated to be proven a working model from the Scyther formal analysis. After that, an informal testing was conducted to test the implementation of the proposed protocol. Though, there was some pitfalls and compromises, a working prototype was developed that proves the feasibility of DRIP. DRIP is a strong security oriented UAS RID that focuses on protecting the users' PII's whilst also meeting the accessibility requirements for observers and authorised bodies to lookup drones. This is possible due to the implementation of registries. This project successfully implemented a working condition registry alongside an API that responds and allows for entities to register themselves and 'go live'. DRIP promotes situational awareness (SA) due to its nature in requirements, and it has proven itself to protect against attack vectors that are consequential such as man-in-the-middle attacks, replay attacks, and impersonation attacks. The future technology of drones relies on the success in researching security-oriented proposed protocols, and DRIP has the potential to become the standard of Network RID and Broadcast RID.

10. Further Research

The topics covered in this research were mostly about UAV-to-infrastructure and UAV-to-observer communication. The next steps to take to explore for a more in-depth study about UAS RID's is exploring UAV-to-UAV communication which opens more doors for innovation in terms of drones communicating to each other automatically. This is a great use case for the future of drones as this can allow for features like automatic DAA (detect and avoid) with advanced path intersection algorithms and for services such as current weather forecast updates to communicate to nearby drones so that they can amend their operation's flight plan to avoid dangerous conditions.

Another area for further study could be the experimentation of multiple drones on a drone network; this can expose different types of vulnerabilities that are better addressed early. Performance testing under different loads to simulate different times of the day/year to simulate real-case scenarios is important – especially with the rapid adoption of UAS RID due to the regulation updates.

References

Abdelmaboud, A. (2021). The Internet of Drones: Requirements, Taxonomy, Recent Advances, and Challenges of Research Trends. *Sensors*, [online] 21(17), p.5718. doi:https://doi.org/10.3390/s21175718.

Agrawal, A. (2019). *Agile Methodology: Incremental and Iterative way of development*. [online] Medium. Available at: https://medium.com/@ashutoshagrawal1010/agile-methodology-incremental-and-iterative-way-of-development-a6614116ae68 [Accessed 14 Apr. 2023].

Ahokas, J. and Persson, J. (2022). *Formal security verification of the Drone Remote Identification Protocol using Tamarin*. [online] DIVA. Available at: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1671102&dswid=-7276 [Accessed 24 Apr. 2023].

Amazon Web Services, Inc. (2023). *What is an API? - Application Programming Interfaces Explained - AWS*. [online] Available at: https://aws.amazon.com/what-is/api/ [Accessed 10 Apr. 2023].

Beekman, J. (2022). *What Is Internet of Drones (IoD)? - IoT Marketing*. [online] IoT Marketing. Available at: https://iotmktg.com/what-is-internet-of-drones-iod/ [Accessed 28 Mar. 2023].

Caa.co.uk. (2023). *EASA part SPO frequently asked questions | Civil Aviation Authority*. [online] Available at: https://www.caa.co.uk/commercial-industry/aircraft/operations/types-of-operation/part-spo/easa-part-spo-frequently-asked-questions/#:~:text=Operator,or%20one%20or%20more%20aerodromes.%E2%80%9D [Accessed 10 Apr. 2023].

Card, S.W., Wiethuechter, A., Moskowitz, R. and Andrei Gurtov (2022). *RFC 9153: Drone Remote Identification Protocol (DRIP) Requirements and Terminology*. [online] IETF Datatracker. Available at: https://datatracker.ietf.org/doc/rfc9153/ [Accessed 8 Apr. 2023].

Cho, G., Cho, J., Hyun, S. and Kim, H. (2020). SENTINEL: A Secure and Efficient Authentication Framework for Unmanned Aerial Vehicles. *Applied Sciences*, [online] 10(9), p.3149. doi:https://doi.org/10.3390/app10093149.

Cremers, C. (2014). *Scyther tool*. [online] Cispa.io. Available at: https://people.cispa.io/cas.cremers/scyther/#:~:text=Scyther%20is%20an%20automated%20security,of%20all%20possible%20protocol%20behaviours. [Accessed 28 Apr. 2023].

Dispatchtrack.com. (2020). *Drones in Logistics and Supply Chain: Uses and Advantages*. [online] Available at: https://www.dispatchtrack.com/blog/drones-logistics-supply-chain [Accessed 8 Apr. 2023].

Drone Pilot Ground School. (2019). *What is the difference between AGL and MSL? - Drone Pilot Ground School*. [online] Available at: https://www.dronepilotgroundschool.com/kb/what-is-the-difference-between-agl-and-msl/#:~:text=Above%20Ground%20Level%2C%20or%20AGL,in%20order%20to%20calibrate%20altitude. [Accessed 10 Apr. 2023].

Europa.eu. (2016). *What is GNSS?* [online] Available at: https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss [Accessed 10 Apr. 2023].

Faa.gov. (2017). *Unmanned Aircraft System Traffic Management (UTM) | Federal Aviation Administration*. [online] Available at: https://www.faa.gov/uas/research_development/traffic_management [Accessed 28 Mar. 2023].

Faa.gov. (2019). *UAS Remote Identification | Federal Aviation Administration*. [online] Available at: https://www.faa.gov/uas/getting_started/remote_id [Accessed 28 Mar. 2023].

Faa.gov. (2022). *Timeline of Drone Integration | Federal Aviation Administration*. [online] Available at: https://www.faa.gov/uas/resources/timeline#:~:text=August%2029%2C%202016,effective%20date%20of%20part%20107. [Accessed 28 Mar. 2023].

Gps.gov. (2023). *GPS.gov: GPS Overview*. [online] Available at: https://www.gps.gov/systems/gps/ [Accessed 10 Apr. 2023].

Hashem, Y., Elmedin Zildzić and Andrei Gurtov (2021). Secure Drone Identification with Hyperledger Iroha. *Design and Analysis of Intelligent Vehicular Networks and Applications*. doi:https://doi.org/10.1145/3479243.3487305.

Ibm.com. (2014). *IBM Documentation*. [online] Available at: https://www.ibm.com/docs/en/SSB23S_1.1.0.2020/gtps7/s7symm.html [Accessed 20 Apr. 2023].

Ibm.com. (2021). *IBM Documentation*. [online] Available at: https://www.ibm.com/docs/en/ztpf/1.1.0.15?topic=concepts-public-key-cryptography [Accessed 20 Apr. 2023].

Insider Intelligence (2023). *Drone market outlook in 2023: industry growth trends, market stats and forecast*. [online] Insider Intelligence. Available at: https://www.insiderintelligence.com/insights/drone-industry-analysis-market-trends-growth-forecasts/ [Accessed 8 Apr. 2023].

Jewett, R. (2023). *How Rapid Growth in Drone Use and EU Regulations Will Accelerate Demand for Satellite Connectivity*. [online] Via Satellite. Available at: https://www.satellitetoday.com/opinion/2023/03/31/how-rapid-growth-in-drone-use-and-eu-regulations-will-accelerate-demand-for-satellite-connectivity/ [Accessed 8 Apr. 2023].

Kashif Naseer Qureshi, Muhammad Perwaiz Sandila, Pu Chun Ke, Tiziana Margaria and Aslam, L. (2022). Authentication scheme for Unmanned Aerial Vehicles based Internet of Vehicles networks. *Egyptian Informatics Journal*, [online] 23(1), pp.83–93. doi:https://doi.org/10.1016/j.eij.2021.07.001.

Ko, Y.-H., Ji Yoon Kim, Daniel Gerbi Duguma, Philip Virgil Astillo, You, I. and Pau, G. (2021). Drone Secure Communication Protocol for Future Sensitive Applications in Military Zone. *Sensors*, [online] 21(6), pp.2057–2057. doi:https://doi.org/10.3390/s21062057.

Lei, Y., Zeng, L., Li, Y., Wang, M. and Qin, H. (2021). A Lightweight Authentication Protocol for UAV Networks Based on Security and Computational Resource Optimization. *IEEE Access*, [online] 9, pp.53769–53785. doi:https://doi.org/10.1109/access.2021.3070683.

MacAskill, E. (2009). *US drones hacked by Iraqi insurgents*. [online] the Guardian. Available at: https://www.theguardian.com/world/2009/dec/17/skygrabber-american-drones-hacked [Accessed 9 Apr. 2023].

ResearchGate. (2018). *Figure 1: ATM , red color, is provided above the local obstacle level,...* [online] Available at: https://www.researchgate.net/figure/ATM-red-color-is-provided-above-the-local-obstacle-level-and-down-to-the-surface-in_fig1_326468351 [Accessed 28 Mar. 2023].

Skybrary.aero. (2014). *Situational Awareness | SKYbrary Aviation Safety*. [online] Available at: https://www.skybrary.aero/articles/situational-awareness#:~:text=Put%20simply%2C%20situational%20awareness%20(SA,aircraft%20%2D%20is%20taken%20into%20account. [Accessed 10 Apr. 2023].

Skybrary.aero. (2021). *Air Traffic Management (ATM) | SKYbrary Aviation Safety*. [online] Available at: https://skybrary.aero/articles/air-traffic-management-atm [Accessed 10 Apr. 2023].

Unmanned Systems Technology. (2022a). *Beyond Visual Line of Sight | BVLOS Technology, Communications for Drones*. [online] Available at: https://www.unmannedsystemstechnology.com/expo/beyond-visual-line-of-sight/#:~:text=BVLOS%20(Beyond%20Visual%20Line%20of,line%2Dof%2Dsight%20flying. [Accessed 10 Apr. 2023].

Unmanned Systems Technology. (2022b). *Sense & Avoid Technology | Detect & Avoid Systems for Drones & Robotics*. [online] Available at: https://www.unmannedsystemstechnology.com/expo/sense-avoid-systems/#:~:text=Sense%20and%20Avoid%20(SAA)%20or,lines%2C%20birds%20and%20other%20obstacles. [Accessed 10 Apr. 2023].

Usemynotes (2022). *Message Authentication Code (MAC) - Use My Notes*. [online] Use My Notes. Available at: https://usemynotes.com/message-authentication-code-mac/ [Accessed 20 Apr. 2023].

Wang, L., Chen, Y., Wang, P. and Yan, Z. (2021). Security Threats and Countermeasures of Unmanned Aerial Vehicle Communications. *IEEE Communications Standards Magazine*, [online] 5(4), pp.41–47. doi:https://doi.org/10.1109/mcomstd.0001.2000078.

Wikipedia Contributors (2022). *List of unmanned aerial vehicles-related incidents*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/List_of_unmanned_aerial_vehicles-related_incidents [Accessed 9 Apr. 2023].

Wikipedia Contributors (2023a). *Advanced Encryption Standard*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard [Accessed 20 Apr. 2023].

Wikipedia Contributors (2023b). *JSON*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/JSON [Accessed 28 Apr. 2023].

Wikipedia Contributors (2023c). *RSA (cryptosystem)*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/RSA_(cryptosystem) [Accessed 20 Apr. 2023].

Wikipedia Contributors (2023d). *Unmanned aerial vehicle*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle [Accessed 30 Mar. 2023].

Wrike (2022). *What is Agile Methodology in Project Management?* [online] Wrike. Available at: https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/.

www.DroneTraining.co.nz. (2020). *What is visual line of sight as defined by CAA (in real world terms)?* [online] Available at: https://www.dronetraining.co.nz/questions/drone-law/what-is-visual-line-of-sight-as-defined-by-caa/#:~:text=Visual%20line%20of%20sight%20means%20a%20straight%20line%20along%20which,to%20correct%20any%20subnormal%20vision [Accessed 10 Apr. 2023].

Appendix

[1] Project code base | GitHub link: https://github.com/Coolide/DRIP

[2] Project sprint deliverables | GitHub: https://github.com/Coolide/DRIP-updates