FFI-RAPPORT

20/01289

Cybersecurity and cryptographic methods in unmanned systems

– a study of the current state in unmanned aerial vehicles and similar systems

Jan Henrik Wiik

Cybersecurity and cryptographic methods in unmanned systems

– a study of the current state in unmanned aerial vehicles and similar systems

Jan Henrik Wiik

Norwegian Defence Research Establishment (FFI) 24 June 2020

Keywords
Kryptografi
Ubemannede luftfarkoster (UAV)
Informasjonssikkerhet
Cyberdomenet

FFI report
20/01289

Project number
1472

Electronic ISBN
978-82-464-3266-3

Approvers
Nils A. Nordbotten, *Research Manager*
Ronny Windvik, *Research Director*

The document is electronically approved and therefore has no handwritten signature.

Summary

In this report, we present common attack methods against unmanned systems and the cryptographic solutions that are commonly implemented to address these issues. We also detail many concrete studies of unmanned aerial vehicles (drones) and their cryptographic solutions and general cyber security.

Our goals with this work are twofold. Our first and main goal is to get a representative picture of the current state of cryptography and information security in available unmanned systems. We do so by investigating known methods of attack against unmanned systems, as well as studying the vulnerabilities and security mechanisms of specific devices. This overall picture of vulnerabilities and cryptographic solutions is crucial to build an understanding of the practical security of modern unmanned aerial vehicles. Through this understanding of the state of practical security we can get an improved understanding of how to handle the operational risk inherent in using unmanned systems, which will be crucial to inform how these systems should be integrated into operations for civilian, commercial, industrial, or military use.

Our second goal is to help bridge the gap between the field of unmanned systems and the field of cryptography. The report highlights many of the practical challenges in security that unmanned systems face today, with the explicit goal of making these challenges more visible to the security community. Similarly, the report highlights which cryptographic mechanisms already exist and are well-understood, and connect them to the vulnerabilities they address, with the goal of increasing awareness of these risks and tools to people well versed in autonomy and unmanned systems. As a result, we hope the two fields can come together to find practical solutions and expose core issues that neither field would naturally discover in an isolated setting.

We start our report by defining the terminology with which we discuss cyber security and unmanned systems in the report. In doing so, we define our assets and adversaries, and which security goals we want to achieve for unmanned systems.

Then, we present common ways to attack the information security of unmanned systems. We divide the attacks based on their domain: hardware, side channels and fault injection, software, and communication. We also make mention of several attacks that utilize multiple domains, as well as other attacks that fall outside of these categories and our scope.

Next, we present cryptographic mechanisms that we have found to be typically implemented on common commercial unmanned aerial vehicles, and how they relate to the vulnerabilities discussed in the report.

Finally, we discuss a series of devices that have been examined by security researchers and enthusiasts, and the relevant work done to understand the security of these devices and how it can be improved.

Sammendrag

I denne rapporten presenterer vi vanlige angrepsmetoder mot ubemannede systemer og de kryptografiske løsningene som er typisk implementert for å møte angrepene. Vi presenterer også flere studier rundt droner og deres kryptografiske løsninger og generell cybersikkerhet.

Vi har to mål med dette arbeidet. Det første og viktigste målet er å kartlegge status av tilstanden til eksisterende kryptografi og informasjonssikkerhet i åpent tilgjengelige ubemannede systemer. Vi gjør dette gjennom å undersøke kjente angrepsmåter mot ubemannede systemer, samt å studere svakhetene og sikkerhetsmekanismene i konkrete enheter. Dette overordnede bildet av svakheter og kryptografiske løsninger er veldig viktig for å få innsikt i den praktiske sikkerheten til moderne droner. Gjennom denne kjennskapen til den praktiske sikkerheten kan vi arbeide med å øke kunnskapen om den operasjonelle risikoen ved å bruke ubemannede systemer. En slik kunnskap vil være avgjørende for hvordan disse systemene bør bli integrert i sivile, kommersielle, industrielle eller militære operasjoner.

Vårt andre mål er å hjelpe med å bygge bro mellom studiet av ubemannede systemer og studiet av kryptografi. Rapporten peker spesielt på mange av de praktiske utfordringene i sikkerhet som ubemannede systemer står ovenfor i dag, med et eksplisitt mål om å gjøre disse utfordringene synlige for sikkerhetseksperter. I samme tråd tar rapporten frem hva slags kryptografiske mekanismer som allerede eksisterer og er godt forstått, og drar bindingen til sårbarhetene de dekker, med mål om å øke bevisstheten rundt disse risikoene og verktøyene for folk som kan mye om autonomi og ubemannede systemer. På en slik måte håper vi fagfolk fra de to feltene sammen kan finne praktiske løsninger og eksponere kjernen til problemstillingene på en måte som de ikke naturlig ville oppdaget hver for seg.

Vi starter rapporten med å definere terminologien som vi bruker for å diskutere cybersikkerhet og ubemannede systemer i rapporten. Mens vi gjør dette definerer vi våre verdier og trussel-aktører, og hva slags sikkerhetsmål vi har lyst til å oppnå for ubemannede systemer.

Så presenter vi vanlige måter å angripe digital sikkerhet for ubemannede systemer. Vi deler angrepene inn i kategorier basert på hva slags domene som benyttes: maskinvare, sidekanaler og feilinjisering, programvare og kommunikasjon. Vi nevner også flere angrep som utnytter mer enn ett domene, og andre angrep som faller utenfor disse kategoriene og rapportens omfang.

Deretter presenterer vi kryptografiske mekanismer som vi har funnet typisk implementert i vanlige kommersielle droner, og hvordan de relateres til sårbarhetene diskutert i rapporten.

Til slutt tar vi for oss en rekke konkrete enheter som har blitt studert av sikkerhetsforskere og entusiaster, og det arbeidet som er gjort for å forstå sikkerheten i disse enhetene og hvordan den kan bli forbedret.

Contents

1 Introduction

The integration of computer technology into every level of modern society is dramatically affecting how we interact with the world around us. One of the features distinguishing this technological revolution from any other is the ability for our newfound systems to act in some autonomous fashion; they are programmed with instructions that enables them to perform tasks with different degrees of human intervention. Some are manually operated in a physical fashion – like mobile phones, or our current generation of cars. Others are controlled manually in a remote fashion– like a satellite, or a drone. Others still are designed to operate without any human input at all – like a proper self-driving car, or an online recommendation algorithm.

Many of these systems and devices have additional commonality in that they have both a physical aspect and a cyber-aspect, and their physical capabilities heavily interplays with their cyber capabilities. As such, they are commonly referred to as *cyber-physical systems* (CPSs). They all contain large amounts of autonomous features, making them generally simpler to operate or enabling them to operate for periods of time without human input at all.

It is clear that this technology is quickly becoming deeply integral to our everyday life and will define future directions in all parts of society. Understanding CPSs – and unmanned systems in general – will therefore naturally be instrumental in planning the future offensive and defensive capabilities of a nation during peacetime, crisis, and wartime.

In this report, we focus on security for certain unmanned systems. It tends to be the case that emerging technologies focus on functionality, commercial viability, and user accessibility more so than security. Unmanned systems appear to follow the same trajectory – despite great advances in their functional capabilities, a thorough and mature understanding of their security and security challenges is seemingly far away. This can be due to a large combination of circumstances. One such circumstance is that a mature understanding of security is something that requires a substantial amount of work. By nature the field is conservative in its progression, as one missed step is all it takes to lose all notion of security. Another is that pursuing good security might not be good business for most commercial actors. Security is simply not something that the majority of customers value in terms corresponding to the expenses required to address it by the company. In other words, it can often be cheaper for a manufacturer or retailer to simply replace compromised units, and leave matters to law enforcement and insurance companies. A third reason is that security in complex systems can be an extremely difficult task due to the inherent difficulty of understanding their composition. From the multitude of different hardware platforms to the millions of lines of code across thousands of files that constitute an operating system, to the physical routing of data between connection points and the individuality of each end user – the system and its attack surface is vast, and requires aspects of several scientific fields to understand in its entirety. These and other circumstances makes security lag behind the development of other features in emergent technologies.

To elaborate on the last point, security is an inherently difficult field to operate in because the subject is inherently oriented towards *systems* rather than *components*. Digital security need to be constructed on a complete, technical, and detailed understanding of the system as a whole. However, for many modern systems there are simply too many moving parts and individual interactions to understand the system at the level necessary for a thorough security analysis. This is especially true for CPSs. Not only would a security analyst have to consider threats to the part of the system operating in the cyber-domain, the analyst would also have to consider threats to the physical parts. Moreover, due to the interplay between the two, the analyst would also have to consider how a compromise of the cyber security properties damage the cyber capabilities, how this would further impact the physical capabilities, and vice versa. In all, this melds together to form a very complex picture.

1.1 Autonomous systems and our scope

The meaning of the terms *autonomy* and *autonomous* is discussed at some length in [1], and ultimately refers to the ability of a system to act without outside influence and input. As such, an autonomous system has some ability to infer what situation it is in and how to respond to that situation in an appropriate way. These situations and responses can have different levels of complexity, reflected in different levels of autonomy for the system. For example, an autonomous robotic vacuum cleaner has a very simple set of situations it is expected to be able to respond to, and its responses are neither particularly complex nor particularly critical in nature. In contrast, an autonomous military land vehicle might encounter extremely complex and critical situations and might have to provide immediate complex and precise responses.

Many unmanned systems have some degree of autonomous capability. This is one of the features that separate these systems from more traditional information systems. A drone that is being remotely controlled is not responding to its situation in an independent fashion, and as such is not acting autonomously. As is discussed in the report by Mancini et al., NIST puts this in a framework called “contextual autonomy capability” [2], which describes autonomy as being parametrised by three properties: mission complexity, environmental complexity and human independence.

We will not concern ourselves with the mechanisms behind autonomous behaviour *per se* in this report, as that falls squarely outside the field of security. However, the unmanned systems we look at will have varying degrees of autonomous capability, and to understand that feature is important to understand the context within which we want to secure the systems as a whole. A particularly interesting class are unmanned and autonomous vehicles, which we can divide according to their environment as *unmanned aerial vehicles (*UAVs*), unmanned ground vehicles (*UGVs*), unmanned surface vehicles (*USVs*),* and *autonomous underwater vehicles (*AUVs*)*. These vehicles could employ anything from lower degrees of contextual autonomous capability to the highest degrees, according to its purpose.

This report keeps this larger context in mind when discussing security and cryptography for unmanned systems. However, we have chosen to focus on UAVs as a primary type of unmanned system that we want to study closer. UAVs are commonly referred to as drones, and are by far the most visible and established unmanned and/or autonomous vehicle. The strict context of UAVs still span a great many devices, as we see in chapter 5, and is a way to explore details around their vulnerabilities and security measures that represent general trends and issues with unmanned systems. Thus, most if not all of what we discuss in relation to UAVs are relevant for other unmanned systems, particularly unmanned and autonomous vehicles.

In our work, we originally wanted to look at military systems as well as civilian ones. However, due to the general lack of open information surrounding military systems and for public accessibility, we are mainly presenting civilian systems in this report. There are some information available regarding military systems, but as the reader of this report will understand this openly available information provides little to no actual insight.

1.2 Securing a system with cryptography

Like for many other complex issues, security is made more approachable by carefully compartmentalising it into smaller aspects that can be studied separately, with clearly defined interfaces. One way to do this is through the OSI reference model [3], which divides networks into seven layers that can be studied separately. When considering CPSs we need to pay special attention to the physical layer of the OSI model compared to many other systems, simply because their physical capabilities are integral in what they do. In particular, we need to consider the attack surface on this layer and what measures can be taken, just like we consider the attack surface and measures on other layers of the model. This is a rather substantive task and not the focus of this report. However, we keep the reference model in mind when assessing the value of given security measures, and what vulnerabilities remain.

The field of cryptography deals with the study of how two parties might achieve and preserve secure communication even in the presence of an adversarial third party. This involves the study of mathematical methods that in some way transform data in clever ways, often relying on the existence of private or public strings of bits - *keys*. These mathematical methods are often referred to as encryption algorithms, though it is more precise to refer to them as cryptographic mechanisms – as we show throughout the report, cryptography is more than just encryption, and cryptographers use tools beyond just algorithms.

One pitfall that must be avoided is *security through obscurity*: the reliance on keeping a design or implementation hidden for it to be kept secure. It sometimes occurs that a protocol or system is wrongly considered secure only or mainly because the particulars of its design or implementation is assumed not to be available for adversaries. In cryptography, this is formulated more explicitly as *Kerckhoff's principle:* A cryptographic scheme should be secure in the situation where everything about the system except one secret string of bits referred to as the *key* is public knowledge. This makes the issues outlined initially all the more crucial to get right – we must get a proper understanding of our own system and architecture and its attack surface, because we must not rely on our adversary not finding an existing weakness simply because we have not been able to find it ourselves. This principle is all the more relevant in the context of autonomous systems, whose autonomic capability render them much more exposed to capture than traditional stationary systems.

Cryptographic mechanisms often aim to provide a promise of security through a logical step called a *security reduction*. The steps in such a reduction are conceptually quite simple. First, assume there is some very hard problem that no one in the world is able to solve efficiently[1]. Then, create a way to handle data so that in order to gain access to the protected parts, a potential adversary would have to solve the very hard problem that no one in the world is able to solve. This way, they would have an insurmountable task ahead of them if they want to access protected data. The results from this method are algorithms that have deep ties to mathematics, such as the ones we review in chapter 4.

The above classic strategy looks very convincing at first glance; however, it does run into a few critical challenges when used to implement security. The first challenge is that there is no known way to be sure that a problem cannot be solved efficiently. There is always a possibility that there exists some new, ingenious method that can solve such a problem in a way faster than previously thought possible. In addition, as computational resources evolve so does our ability to address computational challenges. One famous example is factorization – as computational power and our understanding of number theory has dramatically improved over the last several decades, our ability to factor large numbers have similarly increased. This has had very real implications for information security since much of modern practical cryptography is built on the *RSA cryptosystem*, whose hard problem is precisely factorization[2].

Another important challenge that must be considered when using cryptography to secure information is that the derived mathematical security properties are only a single step in a very long and complex chain of dependencies, all of which need to be in place before any practical security is achieved. These steps in the context of computer systems would include:

- The mathematical design of the cryptographic mechanism
- The implementation of the cryptographic mechanism in software
- The integration of the cryptographic implementation into software solutions
- The design, production and deployment of the hardware that the software will run on
- The deployment of these software solutions into hardware

[1] When we talk about “efficient” algorithms in cryptography and computer science, we typically refer to algorithms that can be solved in *polynomial time* on a typical computer, meaning the task does not grow exponentially in difficulty. It is slightly removed from reality to claim that anything that does not grow exponentially is quick and easy to solve, but that is a helpful way to regard it from a theoretical point of view.
[2] More precisely, factorization of large numbers would yield an attack on the RSA cryptosystem, but it has never been proven that an attack on the RSA cryptosystem is *sufficient* to factorize large numbers. Thus, the RSA cryptosystem is never more secure than number factorization.

- The operational context of the unmanned systems
- The maintenance of the software and hardware after deployment

All these steps contain multiple points of failure that can completely invalidate any security properties we might have started with in our design of the cryptographic mechanism. We will examine several of these steps in much more detail in chapters 3 and 5, where we discuss common attack patterns, the measures taken to protect them in commercial systems, and their vulnerabilities.

The important point to be extracted from this discussion is that although cryptography can be the root of a secure implementation, it does not guarantee in any way or form that the security is propagated all the way to the end user. Every single step in the chain need to be examined and carefully handled before any such security is reached, and we will see many examples in chapter 5 of systems that are completely insecure because they do not take a thorough approach. Cryptography must not be conceived to be a one-step solution to such a complex problem.

1.3 Quantum computers and the “quantum threat” to cryptography

Since cryptographic security is so heavily tied to computational efficiency, technological advances affect our view of cryptographic security. Over the last decades, this has become apparent in recommended key sizes for many cryptographic protocols. In the current field, current and projected future advances in quantum physics and technology based on quantum properties make quantum processors a generally accepted legitimate threat to cryptography for the next decade. The core of the issue is that cryptographic protection is designed to last for decades – which necessarily implies that the field needs to anticipate the technological advances during that future period. This is by no means an exact science, and the community can only make projections based on the worst case[3] scenario and make sure future systems and algorithms are equipped to handle that.

The possible introduction of for quantum computers in the coming decade, with some projections explicitly pointing to the technology being possible within 2030 [4] [5], is today primarily a pressing issue for confidentiality protection. This comes naturally from the timeline of confidentiality protection – information we are protecting *today* is being encrypted with the goal of keeping the information inaccessible for several decades into the future. Therefore, confidentiality can potentially be violated for information exchanged today. This is not such a pressing issue for integrity protection, since the ability in a decade to forge a current day message is less of a threat. However, this by no means mean that one should not address the issue of integrity protection – we still need many years to standardise and implement efficient and secure cryptographic algorithms that are not susceptible to quantum processor attacks like

[3] Or best case, if seen from the perspective of a technology optimist.

Shor's algorithm [4] and *Grover's algorithm* [5]. We will not cover these in this report, and instead refer to the references for a deeper look into the topic.

For the remainder of this report, we will comment when solutions we find are not considered quantum secure. However, we will only consider this a potential security vulnerability in the context of confidentiality protection.

1.4 Report purpose and structure

This report serves a dual purpose. Firstly, it is an effort to outline relevant challenges and tools in security to the community studying unmanned systems. Raising awareness of these issues and the measures that already exist to tackle them goes a long way in making sure technology is implemented in a secure way. It is also an effort to outline relevant unmanned systems and their usage to the community studying computer security. In bridging the gap between the two communities, we hope to provide some common ground enabling discussions about the security aspect of using unmanned systems, and which challenges are inadequately addressed by existing solutions. This is especially important in the context of critical applications such as governmental operations, industrial operations, and military operations.

The report is divided into chapters in the following manner: In chapter 2, we provide an overview of, and general classification of, available autonomous vehicles that could be considered potentially viable for some military operation. In chapter 3, we present known attacks against such vehicles and similar systems that pose potential threat to the systems for such operations. In chapter 4, we review cryptographic mechanisms that are implemented in the systems presented in chapter 2 in order to defend against the threats presented in chapter 4. In chapter 5, we discuss which security mechanisms would constitute a sufficient level of protection and outline some ideas of how this can be organized. Finally, chapter 6 collects these thoughts and present our conclusions around the security level of autonomous vehicles as well as their suitability for critical applications.

2 Definitions and assumptions

The purpose of this chapter is to establish terminology and a mindset suitable to discuss the problems which the rest of this report focuses on. This is of crucial importance in security to facilitate precise discussions and understanding. After such an initial frame of reference is established, we can properly begin to understand how the cryptographic measures fit into the bigger picture of securing digital technology. This is the objective of this chapter. Most of the notions are applications of general concepts in security, and as such, this chapter lies firmly in the area of introducing security to non-security professionals.

We begin by presenting the *assets* that we want to secure in an unmanned system. We then move on to specifying which objectives or *security goals* we want to achieve in order to call the system secure. Following that, we discuss which kind of adversaries we want to secure our system against and their possible capabilities. We put this in a broader context within which we discuss attacks and security mechanisms, as well as a discussion of what we want to protect. This also serves to solidify the scope of the work in this report. We then present the most common and most important attacks whose aim is to compromise information security, in the context of unmanned systems. These attacks are categorised based on the domain with which they target. Parts of this work is reflective of our work within [1] and we direct interested readers there for further information.

2.1 Assets

The most important pieces in our framing are properly defining what we are looking to secure, and what it would mean for it to be secure. The former is referred to as *assets,* which we discuss in this section, and the latter we address in the next section. An asset in a general sense can be anything that is deemed valuable, such as a bank account, a car, a telephone line, an email password, or coordinates for a military base.

In this report, we are looking at assets of a purely digital nature. That is, we are interested in measures taken to secure *communication*, *stored information*, and *software*. This is in line with what cryptography typically concerns itself with as a field – cryptographic solutions in isolation do not address, and would not prevent, say, an armed robbery or the cutting of an internet cable. Despite being of a strictly non-physical nature, it is important to note that compromise of digital assets can yield outcomes equally terrible as that of other assets. Unauthorized access to data, identity theft, and the denial of vital capabilities can quickly escalate to leakage of damaging information, monetary loss or even the loss of lives.

It is important to appreciate that although the goal we concern ourselves with in this report is securing digital assets, we should not be ignoring physical aspects of the system. The physical systems involved in managing the digital assets do still play a very important role in our discussion, as we cannot secure software without first having some control over the hardware it is running on – we will see examples of this later in the report.

2.2 Security goals: the CIA triad

We use a very common security perspective, through the widely adopted *CIA triad* of information security goals. These goals are used throughout the fields of cryptography and security, and might be familiar to the reader.

The first information security goal we want to achieve is *confidentiality*. The Federal Information Security Management Act[4] (FISMA) defines confidentiality as *"preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information"*. It is, in short, the protection of unauthorized access to information. Confidentiality is the objective most commonly associated with cryptography and encryption, and the algorithms that seek to achieve this objective are *encryption algorithms*. Encryption algorithms work by utilizing some known method to scramble information into a state that is indistinguishable from random noise to an unauthorized party observing the message. Only someone who knows the predetermined secret (such as a key) can efficiently unscramble the message and reveal the intended information. The most common confidentiality-focused algorithms for unmanned systems are presented in chapter 4.1.

The second information security goal that we want to achieve is *integrity*. FISMA defines this objective as *"guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity"*. In short, it is the security objective that no unauthorized party can change the information or messages, nor can they send fake messages. The latter property in isolation is what is commonly referred to as *authenticity*, as in the FISMA definition. They also mention the property of *nonrepudiation*, which refers to the idea that once a message has been sent, the sender has no possible deniability. In other words, there cannot be any sort of retraction where the sender claims it never did send the message. As such, if there is no authenticity (for example) there is no nonrepudiation, since the sender can always claim that someone else sent the messages. There are many algorithms used to obtain integrity, as we explore more in chapter 4.3. Often they are part of an *authenticated encryption algorithm*, referring to an algorithm that provides both message confidentiality and integrity.

The third and final information security goal is that of *availability*. The FISMA definition of availability is that it is *"ensuring timely and reliable access to and use of information"*. In essence, it is the assurance that information is flowing correctly. It is important to understand that cryptography does not in essence work to *obtain* availability; rather, the focus is to *maintain* availability to those authorized for it, whilst restricting it for all others. Furthermore, in contrast to the other properties, availability is not a property that you either have or you lack; you will usually have degrees of availability. Some information might require a working internet connection to access, whilst others might even only be available at given physical locations. Cryptographic algorithms contain some degree of overhead, which is also a form of reducing availability. We make mention of this when relevant in our discussion of the algorithms in chapter 4.

[4] The FISMA act can be read in its entirety at https://csrc.nist.gov/topics/laws-and-regulations/laws/fisma.

We refer to steps taken to achieve these security goals as *security mechanisms*. If these mechanisms are of a cryptographic nature, such as cryptographic algorithms, we can also call them *cryptographic mechanisms.*

2.3 Adversaries

In this chapter, we have so far discussed what one might want to secure in unmanned systems and what it would mean to secure them. This section expands upon this by defining *whom* we want to secure our system against in the sense of what they are capable of, and *what* we expect them to be able to do to compromise our security goals.

An *adversary* is an unauthorized party trying to break our security goals. Intuitively, we understand that there are many very different kinds of adversaries which we might want to consider. On the one hand, we might want to make sure our system cannot be hijacked or its communication monitored by a local computer enthusiast. On the other hand, and in particular for applications in military, we might find it of high importance to secure our systems against foreign intelligence agencies and their possible resources. Obviously, these two adversaries require two fundamentally different notions of security. Generalizing from these intuitive examples, we will concern ourselves with two adversary concepts.

The first is an adversary whose resources are generally limited. We assume they have computational power equivalent to that of a small network of modern high-end computers. They have some general knowledge of computer security and hacking, but no system-specific knowledge unless it is openly available on the internet. We assume that they have no method of obtaining physical access to the system in question. We call this adversary a *basic adversary*, and it may describe a relevant potentially threatening individual or a small group of people wanting to interfere with the system.

The other adversary is one whose resources are considerably more potent. We assume they have high computational power, equivalent to advanced supercomputers and large clusters – they may even have future access to a quantum computer, as discussed in section 1.3. They are highly knowledgeable and experienced in cyber security, and can obtain system-specific knowledge on demand. We also assume that they can initiate an operation to gain physical access to the system in question. This includes momentary in an undetectable fashion and prolonged exposure in a controlled environment, like a research laboratory. We call this adversary an *advanced adversary.*

Both types of adversary are important to keep in mind as we review the vulnerabilities found in many unmanned systems in chapter 5. In particular, the advanced adversary is very important when analysing the threat surface against systems that are intended for use in a military setting. We find in chapter 5 that there is minimal – if any – protection against such an adversary, cryptographic or otherwise, on systems that are intended for commercial, non-military use. This must be considered when reviewing the appropriateness of so-called commercial off-the-shelf products for these purposes.

3 Categorizing attacks on unmanned systems

In the last chapter, we presented terminology for discussing security properties and compromises of these, as well as how we can tie cryptography to these properties. In this chapter, we use this language to present the most common and important attacks that can be used to compromise information security in unmanned systems. We also include references to many of the studies we look at in chapter 5, many of which concern the Chinese drone manufacturer DJI due to their apparent popularity among security researchers and enthusiasts as well as consumers.

These attacks are categorised based on the domain which they target. The structure of the categorisation is based on the Common Attack Pattern Enumeration and Classification (CAPEC) community, an openly available[5] online resource that aims to be a "*catalogue of common attack patterns that helps users understand how adversaries exploit weaknesses in applications and other cyber-enabled capabilities*". The attacks we present here are those we have found in literature and similar content that apply to unmanned systems, and are divided into four categories in accordance to their domain of attack;

- *Hardware*, targeting the physical components
- *Side channels and fault injection*, intentionally creating and/or exploiting atypical behaviour and information leakage
- *Software*, targeting the software and its source code
- *Communication*, extracting information from communication channels

There are some attacks that do not fall exactly in one of these categories, and we briefly outline them at the end of the chapter. There are also *hybrid attacks* that make use of several of these domains and exploit a combination of them. We also briefly outline this at the end of the chapter.

One of the ways to attack cyber-physical systems that we mention in this chapter is *reverse engineering* – the process of picking apart how something works to better understand it. We make a clear distinction based on the domain being reverse engineered. Reverse engineering of software is a very common and powerful way to attack a cyber-physical system, and is vital to enable other types of software and communication attacks as many are reliant on knowledge of the target system. We write more about this in section 3.3.2. Similarly, reverse engineering of the hardware might often play an enabling role in many attacks listed here, especially those requiring knowledge of the hardware platform of the system. However, due to its role as

[5] The resource is found at https://capec.mitre.org.

primarily enabling other attacks we will not examine reverse engineering much further in this report.

3.1 Hardware attacks

A *hardware attack* describes an attack where the adversary implements changes to or secretly replaces parts of the hardware configuration of the target system. The adversary could be inserting unauthorized hardware into the configuration, they could be modifying existing hardware through replacements or tinkering, or they could be removing certain hardware components. It is easy to forget that these physical components are where the information is physically processed and stored, as computers ultimately rely on components such as circuitry, ports, and transistors to perform their function. By interfering directly with the physical components, an adversary may diminish or entirely bypass security measures, including cryptographic measures, or they may access information about the system or its stored information which they should not be able to access. It is worth noting that this sort of meddling could take place both after deployment and during manufacturing and delivery – the latter is part of what is referred to as a *supply chain attack*, which we touch upon in section 3.5.

The levels of expertise required to perform a hardware attack vary greatly depending on the attack. Mostly, it does not require the most sophisticated techniques to perform since it operates on a component level, but it will require some knowledge of the hardware configuration of the target system. We consider this mainly a threat from advanced adversaries, including particularly capable private actors, due to its reliance on physical access. It is however worth noting that if the adversary does have physical access, the attack is not particularly complicated to perform. The components themselves could be anything from very cheap and general purpose equipment to quite expensive and custom made solutions. The biggest downside of hardware attacks is that it requires physical access to the target, possibly over an extended period of time -

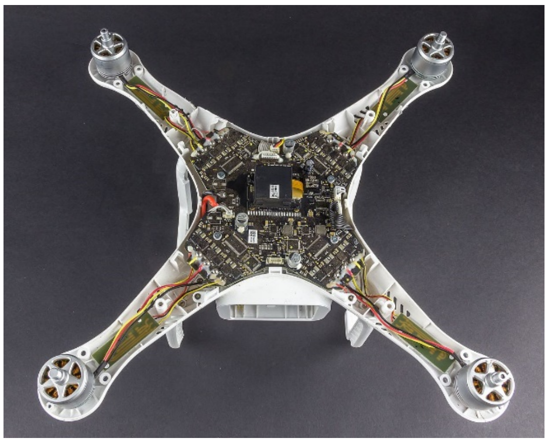


Figure 3.1 The inside of a DJI Phantom III UAV, which has plenty of connections and components to target for an adversary. Photo by Raimond Spekking, under Creative Commons 4.0 license.

an adversary might seek to overcome this by for example accessing the target hardware through faked maintenance routines or before the hardware is deployed.

There are several examples of hardware attacks in literature. In one example we examine more deeply in chapter 5.3, a modified chip was inserted into a UAV control system that allowed an adversary to perform a so-called man-in-the-middle attack [6]. In another instance, researchers managed to connect to a platform in an unintended way through physical manipulation and rewiring of the motherboard, giving access to a port that was intended to be inaccessible [7]. Another paper examined a disassembled DJI Phantom III UAV, disconnecting wires and removing the permanent glue meant to keep the internal storage inaccessible [8].

For unmanned systems used for military operations and other critical applications, hardware attacks might be anything from quite likely or very unlikely, heavily dependent on the nature of the operation. Traditionally, potentially vulnerable computer systems have been kept in controlled areas, guarded by human personnel. One reason for this policy is to protect the systems against hardware-based attacks. However, with the advent of widespread military usage of unmanned vehicles such as UAVs and AUVs this policy can no longer be enforced, and we will have to expect that these vehicles and other systems can be targeted by hardware attacks while being unmanned and unmonitored. All adversaries might attempt to perform such attacks, but the main threat comes from advanced adversaries with extensive resources to investigate the system and inject custom hardware.

Unmanned systems must for these reasons be presumed to be vulnerable to hardware attacks for certain operations and tasks. Measures should therefore be taken to protect against hardware attacks if the systems are to be used for operations where operating in an uncontrolled environment is a likely possibility. Some measures that could be implemented are of a cryptographic nature, such as authentication of components for internal communication and integrity checks of hardware configuration upon boot. However, most measures are not of a cryptographic nature, and typically include things like anti-tamper protective cases and coatings, anti-tamper sensors and more [9].

3.2 Side channels and fault injection

A *fault injection attack* is an attack targeting software or hardware by forcing it into unintended states, making it operate in similarly unintended (i.e. faulty) ways. If done in a controlled manner, the ways in which the software or hardware fails can be exploited to diminish or entirely bypass security measures, or even to extract critical information such as cryptographic keys. Physical faults can be introduced through electric current, glitches, temperature variation, optical faults, or electromagnetic fields. Software faults can be introduced through exploitation of bugs and glitches. Fault attacks can target virtually any part of the system, including input parameters, data processing, storage, or instruction processing [10] [11].

Similarly, a *side channel attack* is an attack targeting unintended information leakage. This leakage can be a normal feature of the system, such as response times or power consumption, or it can be an unintended feature stemming from fault injections. Side channel attacks can themselves be further categorized according to their level of invasiveness, into three levels: *passive*, *semi-invasive*, and *invasive* [11]. Passive side channel attacks target normal features of the system as described earlier, whilst the other two categories involve some amount of fault injection or similar techniques to expose information leakage. Since the two are interoperable and overlapping, we treat them both in this same section.

One feature separating side channel and fault injection attacks is that the majority of techniques require quite sophisticated knowledge and equipment to perform the attack. Most attacks are considered relatively difficult and expensive, and would require quite extended physical access to the system as well as an understanding of how to manipulate the system correctly. Realistically, such attacks would as of today be expected to take place in a reverse engineering and/or forensics lab, and thus is considered an attack mainly reserved for government actors and other advanced adversaries. There are, however, several attacks that do not require such an environment, like software-based glitch exploitation – see the references [10] [11] for an exposition of possible attacks, which include attacks of this nature. These attacks might therefore become more common in the future, even for more basic adversaries.

One rather particular feature of invasive attacks, such as those using fault injection, is that the intrusive attack pattern might permanently damage the system. For physical faults, this might even be visible on the system itself. This can be disadvantageous for the adversary, due to possibly ruining the information stored, ruining other potential ways of extracting information, and ruining any chance of reintroducing the system without being detected.

Similar to our discussion on hardware attacks, systems that are being kept in controlled areas are not in particular danger to most side channel and fault attacks. However, systems operating in uncontrolled environments (and in particular unmanned vehicles) can be exposed to this type of attack, especially through capture. As such, it is a potential vulnerability for systems intended for use within those sorts of operational contexts.

Many measures can be taken to secure against fault attacks, and we will not detail them here. However, they are often quite expensive and involved due to the sophisticated nature of the attack. In addition to physical protection such as protective films or shells, there are measures such as *error correcting codes* [10], *re-keying schemes* [12], and randomization of execution and storage that work to mitigate the potential impact and secure against faults. Anti-tamper protective measures, such as those against hardware attacks, are also effective non-cryptographic measures that can be taken against side channel and fault injection attacks.

3.3 Software attacks

The most well-known form of attacks against any sort of digital system is undoubtedly *software attacks*, meaning attacks where the adversary targets and uses software to breach the targets security goals. Such attacks could be centred on introducing malicious software onto a target platform, modify existing software to behave in ways that benefit the adversary, or access software that otherwise would not be accessible. This is different from fault injection due to the nature of the modifications made – while fault attacks focus on forcing physical faults that change the behaviour of the software, software attacks target the system on a software level. These methods can allow an adversary to gain access to confidential information, change files, change the operation of the system, remove all accessibility to the stored information, and more. Two specific types of software attacks are software injection attacks and reverse engineering attacks, which we will elaborate on below. These can be conceptualized as overcoming system integrity and confidentiality, respectively.

It is worth highlighting exactly how common these sort of attacks are – mistakes and oversights happen all the time in software development, leading to vulnerabilities in the finished software even for the most professional products. There is a large ongoing effort across the world to both find and exploit such vulnerabilities, both for benign and for malicious reasons, and many are found by amateur or semi-professional actors. The Common Vulnerabilities and Exposures list (CVE) contains more than 12,000 vulnerabilities[6] discovered in 2019, over 2500 of which are rated at a high or critical severity, and most of them are relating to software. Many of these are from very established developers and products, such as Google Chrome, Android, Mac OS X, Facebook, and Microsoft Edge.

Due to the prevalence of software vulnerabilities, many companies have so-called "bug bounty"-programmes to incentivise community engagement in exposing such vulnerabilities in a benign manner [13]. Mirroring this, there is also a significant black market for vulnerabilities that are not currently known. The two naturally compete for who gets access first to newly discovered vulnerabilities.

Most of these vulnerabilities do not directly relate to implementations of cryptographic mechanisms, though some do[7]. Instead, they relate to the security of features on their respective platforms. These vulnerabilities may provide backdoor access to read an otherwise protected part of a file system, or give an adversary the power to modify parts of the file system that should have required special authentication. In one instance, an anonymous hacker seemingly exploited an unpatched vulnerability to gain access to classified information about several American surveillance UAVs[8]. It is clear that these vulnerabilities invalidate what security mechanisms – including the cryptography – are intended to achieve, making it highly relevant in relation to their usability.

[6] The list is found in full at https://cvedetails.com. A vulnerability rated as high or critical is a remote access vulnerability giving partial or complete compromise of one or more security properties, where they focus on confidentiality, authentication, integrity, and availability.
[7] One striking example of this was under an open test of the usability and security of electronic elections. The implementation of the cryptographic system was flawed, which led to the security of the system being severely compromised. More about the incident can be found in an (Norwegian) article called "*Feil i krypteringen av e-stemmer*", at https://tu.no/artikler/feil-i-krypteringen-av-e-stemmer/234436.
[8] The WIRED-article is called "*A dumb security flaw let a hacker download US drone secrets*", and can be found at https://wired.co.uk/article/router-drone-reaper-military-secrets.

3.3.1 Software injection

A *software injection attack* or *code injection attack* is a specific kind of software attack where the adversary compromises the behaviour of existing software on the system by injecting adversarial code. There are several ways such an injection could be performed – it could be through a malicious *spoofed* (i.e. faked) software update, a hardware connection such as a USB port, an unprotected communication channel, a stealthy inclusion of unwanted software packaged with other software, or a range of other sources and variants. It includes the sort of attacks that are commonly referred to as malware, such as adware, spyware, and ransomware. The attack surface is very vast for this type of attack, exemplifying exactly how difficult it is to defend against software-based attacks.

Software injection attacks can range from remarkably simple to execute to extremely complex and sophisticated, though it often is much easier to use than to discover. For systems without sufficient security mechanisms, a basic adversary is very capable of performing software injections attacks through some part of its attack surface. This is especially true for systems connected to the internet. We therefore consider UAVs to be potentially vulnerable to this kind of attack, even against a basic adversary.

There are many examples of software injection attacks in the literature on unmanned systems, and it is often reported on by media. In several papers studying the security of established commercial UAVs, the authors are able to inject software through badly protected access ports, either physically or wirelessly [14] [15]. We examine these in some detail in chapter 5.

There are many cryptographic mechanisms that can protect against software injection attacks. Ensuring the integrity and authenticity of incoming software is of particular importance. Cryptographic mechanisms to do so are outlined in section 4.2, and message authentication codes, as well as public key signature schemes. Techniques to ensure the integrity of the software configuration as a whole can also be implemented, especially on smaller systems such as FPGAs.

3.3.2 Software extraction and reverse engineering

A *software extraction attack* or *reverse engineering attack on software* is in some ways the natural opposite of a software injection attack; rather than injecting malicious code, the adversary extracts sensitive software from the target system. This is done using much the same access points as software injection, as discussed in section 3.3.1. Sometimes, the software itself could be sensitive and as such a protected asset on the system, in which case this attack directly violates the security goals. This is for example the case for software containing information giving a competitive advantage, or military classified information. At other times, the extraction of software works as a step in the process of gathering greater understanding of the system and its security mechanisms and potential vulnerabilities. In either case, the adversary will examine the extracted software with tools dedicated to software analysis, possibly over an extended amount of time. The adversary might also modify the software in the process. This can often be followed up with injecting the software back into the system to obtain a desired, modified behaviour.

One well-known instance of reverse engineering is *cracking* or *jailbreaking* of a system or program. In order to obtain a cracked program, an adversary has performed a software extraction attack on the given system and reverse engineered the program, decompiling it and recompiling it with modified behaviour. This is done with the purpose of removing certain restrictions and/or security features from the system, or add extra features that were not intended by the original developer. For example, an adversary might during the reverse engineering process discover precisely at what point a program checks if a given password is correct. The adversary can then remove that part of the file, recompile the file and inject it back onto the target system to modify the behaviour and remove the restriction.

Software extraction attacks and reverse engineering attacks can be performed by any adversary. A key feature is that it primarily relies on the technical expertise of the adversary. Therefore, it is a much dangerous attack when performed by an advanced adversary, which might have access to a great amount of technical expertise and the resources to fund the effort. A reverse engineering attack from such an adversary is therefore a very serious threat even against a heavily secured system, whereas the basic adversary mostly is a threat against more vulnerable systems.

Most of the practical examples discussed in chapter 5 are experiments that start with reverse engineering of software. There are examples where software assets that are considered sensitive by the producer is accessed by the adversary, revealing very sensitive information like developer passwords [14]. Information that by itself would not be considered sensitive is also used in several studies to extract data or enable arbitrary software injection [15] [16]. A special example of reverse engineering is found in modern UAVs. There, developers often include features such as no-fly zones and artificial speed limitations, which some consumers do not enjoy. This creates a market for cracked UAV software that removes these software-imposed restrictions[9].

There are many cryptographic mechanisms that can be employed to secure a system against software extraction and reverse engineering. Mechanisms that ensure the confidentiality of the target file system are the primary tools for this context, and outlined in section 4.1. It is important that this protection also is extended to the files themselves – there are many techniques for obfuscating programs and making them harder to reverse engineer. An extensive study of these can be found in [17].

[9] An example article detailing this phenomenon is found in the article "*DJI drones can get past no-fly zones thanks to this Russian software company*", found at https://theverge.com/2017/6/21/15848344/drones-russian-software-hack-dji-jailbreak. Another article is "*Hackers able to turbo-charge DJI drones way beyond what's legal*", at https://theregister.co.uk/2017/07/11/dji_drones_app_sec/.

3.4 Communication attacks

So far in this chapter, we have described attacks where the adversary targets a single system. A *communication attack*, in contrast, is an attack where the adversary targets a communication channel between several systems, and does not concern itself with the systems themselves. This is in principle a quite well-understood context, since it is the model upon which all of traditional cryptography is built. Still, there are plenty of ways that an adversary can attack unmanned systems through their communication channels, and any adversary could seek to use this domain to attack a system. We divide them here roughly into *sniffing attacks* and *traffic injection attacks*, based on what security goal the adversary overcomes.

There are many different ways the communication channel itself could behave. On the one hand, the channel could be an electromagnetic channel like a Wi-Fi signal or radio link. Such a channel is very well studied, and often have a relatively high bandwidth compared to other alternatives. The channel could even route through the internet or another such complex network of computers – or even a satellite, drastically increasing its range of operation. On the other hand, the channel could be something like an acoustic underwater channel. Such a channel is not as standardised. It would also be much nosier, and it would have much lower bandwidth, making it significantly harder to communicate on the channel and putting strict restrictions on the overhead potentially introduced by security mechanisms. Moreover, some channels could be entirely contained in secure environments, whilst others move through insecure or even compromised environments. For any concrete unmanned system, it is thus very important to keep in mind the nature of its communication channels when considering its security mechanisms.

There are clear similarities between some communication attacks and software injection attacks. Going back to some of our earlier examples of software injection attacks, this is very clear – for example, injecting software through an open port can reasonably be described as a communication attack just as much as a software attack. In this section we describe attacks where the adversary is targeting an active communications link. Still, it is useful to keep in mind that there is a very clear synergy between communication attacks and software attacks.

3.4.1 Sniffing

Perhaps the simplest of communication attacks, a *sniffing attack* consists of the adversary monitoring the channel of communication without the knowledge or consent of the parties involved, and being able to extract sensitive information. This can be critical even on an encrypted communication link, because the adversary might seek to capture the cryptographic key exchange messages as a step to recover the key and thus break the encryption.

There are plenty of examples of sniffing attacks, both in the context of unmanned systems and outside of it. In one example that we examine more closely in chapter 5, commands between a UAV and a central server were sniffed and easily decrypted due to an issue with the cryptographic protection. The communication was then used to reverse engineer commands for the UAV, providing the adversary with complete control of the system. In another example described in chapter 5, Wi-Fi communication during the key exchange step was sniffed and a well-known exploit used to break the encryption.

To secure against sniffing, encrypting the communication channels is the clear cryptographic solution. The communication should be secured with strong cryptographic mechanisms providing confidentiality – such as the AES algorithm presented in chapter 4. This, however, has to also be complemented with good management and distribution of cryptographic keys, as several of the examples we mention above arise as a result of insecure key management.

3.4.2 Traffic injection, spoofing and man-in-the-middle

Rather than just sniffing on communication, an adversary might be interested in inserting their own communication into the channel. Such communication could be fake messages to the systems in question, or more subtle modification of the existing communication. Such an attack is referred to as a *traffic injection attack*. The goal of a traffic injection attack might simply be to reduce the quality of the channel or disrupt the flow of information (i.e. availability)[10], but it could also be used to manipulate the situational awareness of the targets by transmitting false information.

If the adversary manufactures messages that appear to be from another (typically authorized) source, the attack is referred to as a *spoofing attack*. An adversary might decide to use a spoofing attack in conjunction with a sniffing attack to intercept messages, potentially modify them as they please, then transmit them to the intended recipient without them being aware that the communication was intercepted. This is a very famous adversarial model in cryptography and computer security in general, and is referred to as a *man-in-the-middle attack*. Without the right precautions, a man-in-the-middle attack can be very devastating on a communication link or network.

In the context of all unmanned systems, there naturally arises a very particular and powerful attack vector on the communication channel between the system and a potential operator. For autonomous systems, this may or may not be as important as for manually controlled systems, due to their diminished reliance on instructions. Still, there will typically be some communication channel at some point that is a prime target for traffic injection. In many of our practical examples in chapter 5, traffic injection and spoofing is used to take complete control over unmanned systems with some autonomous capability – one such example is the natural continuation of the attack mentioned in section 3.4.1 between a UAV and a server, where commands are injected by the adversary to the UAV pretending to be the server. These commands could be very detailed, like command and control data, or they could be higher-level instructions relying on the autonomous capabilities of the system to be performed.

[10] In the extreme case, injection of a lot of noise would qualify as a jamming attack, which we mention at the end of this chapter.

Another feature that can be important to remember for cyber-physical systems in general (including UAVs) is that such command injections could enable the adversary to move the systems to a location where they cannot be recovered, or at least to get the systems to destroy themselves. These attacks can be completely devastating for the system and some are relatively simple in nature, making them a high priority to address.

Luckily, there are many cryptographic mechanisms created specifically to deal with these attacks. In particular, protection against traffic injection would be provided by integrity and authenticity algorithms like message authentication codes and signatures. Thus, even if the communication is intercepted it can no longer be modified or spoofed. As we discuss in chapter 4, an authenticated encryption algorithm like AES-GCM gives good protection against traffic injection and spoofing whilst also providing security against sniffing, making them an ideal candidate against communication attacks as a whole.

3.5 Additional attacks and security challenges

There are many ways an unmanned system can be attacked that we have not covered so far in this chapter. We have chosen to focus only on those that can be addressed in a (somewhat) satisfactory manner using cryptographic mechanisms. This section provides a small overview over the most important attack vectors an adversary might employ where there are no existing cryptographic mechanisms that address them.

The first such type of attack and the most important one is a *jamming attack*, or similarly a *Denial of Service (DoS) attack* or flooding attack. Jamming is the act of overpowering a signal with another signal that do not contain any useful information, just noise. It is effectively exemplified by how having a conversation is utterly impossible in a very loud environment, such as at a night club or with a third party shouting very loudly right next to you. Similarly, a DoS attack is performed by flooding the target system with meaningless traffic, making it unable to handle and/or recognize the communication that is from the correct source. These are attacks that diminishes availability more so than anything else, and it is challenging to address these issues by cryptographic mechanisms – if you add authentication to incoming messages, for example, you still end up with that overhead for each incoming message.

Another attack that is worth highlighting on its own merit is a *navigation-based attack***.** For such an attack, the adversary targets the communication link of the navigation system, such as a GPS. The adversary then either uses jamming (as discussed above) or uses the techniques in section 3.4. The reason why we do not include this among the other attacks in section 3.4 is that for the vast majority of users and producers, there is no option to change how the navigation signal is transmitted or received beyond a choice of which provider to use. For these users, the problem cannot be addressed. The typical navigation system is not cryptographically secure and vulnerable to spoofing, traffic injection, and eavesdropping – in fact, this has supposedly been used to capture American UAVs in the Middle East, though the truth of the claim is uncertain

[18][11]. Only very resourceful organizations can use cryptographic mechanisms to defend against this by employing their own navigation system, such as employing custom satellites with the mechanisms discussed in section 3.4 implemented.

There are also *supply chain attacks*, in which the adversary targets a supplier of a product at a potentially more vulnerable stage. In essence, it might be more effective to target factories manufacturing UAVs than to target a police force using those same UAVs, and it might be more effective yet to target the company developing the firmware used to control the radio link of the controllers. It would be up to these suppliers to use security mechanisms such as cryptographic mechanisms to secure against attacks.

In addition to the uncategorized attacks listed here, there are other security challenges that an unmanned system will face. They can be targeted by threats through other means entirely than what we have discussed here, like operators being targeted by phishing attempts or other attempts at social engineering. These lie far beyond the scope of cryptography and this report. Also, as we have noted several times in this chapter, there are some very powerful attacks that can arise as a *hybrid attack* using several domains in conjunction. An example is the *rowhammer attack*, which uses injected software to enable fault injection - see [11] for more information on this attack. Rather than addressing them explicitly with cryptographic mechanisms, these attacks should be kept in mind when implementing mechanisms for each domain.

[11] See also our brief mention of the incident in section 5.4.1.

4 Common cryptographic mechanisms

In this chapter, we look at different cryptographic mechanisms we have found to be most implemented in unmanned systems. Like we discussed in chapter 3, there are many attacks on unmanned systems and many ways to mitigate them. The chapter is structured after what security goal each mechanism is addressing.

It is worth noting that for cryptographic purposes, when claiming something is secure we assume a realistic level of computational power and require the probability of discovering a solution to be completely negligible in a long timeframe. For example, a cryptographer would not consider something even close to secure if the chance of finding the solution by pure chance is one in a billion (10^9), as despite each chance being very low modern computers can do billions of operations each second. Instead, today the chance should be multiple orders of magnitude higher – in recent times the standard has been $2^{128} \approx 3.4 \times 10^{38}$, called 128-bit security[12]. Recently many actors opt for an even stronger benchmark, namely one in 2^{256} or 256-bit security.

4.1 Confidentiality protection

The first security goal that we will examine is confidentiality, as presented in section 2.2. The measures we present here are two algorithms we have found to be commonly implemented and supported among unmanned systems and similar systems, called *AES* (Advanced Encryption Standard) and *ChaCha20*. These two are the only two algorithms that are recommended cipher suits in the popular TLS 1.3 protocol[13]. They are both concerned with obfuscating information, scrambling strings of bits to a point where the original information cannot feasibly be recovered without a secret key. They are both symmetric algorithms, meaning that they presume the existence of a key that is shared and known to both parties; the encrypting party as well as the decrypting party, in literature commonly referred to as Alice and Bob respectively.

Despite being superficially similar, the two algorithms have some meaningful differences. AES is what is called a block cipher; it fundamentally works by applying transformations to a string of bits of a given length – a block – which is then included in a larger protocol called a *mode of operation*. This mode is a framework around the algorithm that extends the transformation from one block into an entire message, and there can be several viable choices of mode for any given block cipher depending on its intended usage. ChaCha20 however is a stream cipher, working by producing a sequence of bits seemingly indistinguishable from randomness that can be used directly to obfuscate the message. These differences, as well as differences in the operations they employ, give them unique strengths. In particular, AES is often implemented on systems

[12] A consequence of this approach is that parameters for security should be expected to change over time as technology improves. During World War II, a message would be secure if the protection required trials on the order of 2^{64} – the famous German encryption device called the Enigma machine operated approximately on this order of magnitude. As technology progressed towards the 21st century, this level was no longer sufficient, and new algorithms had to be put in use. Today we face a related issue in regards to quantum computers, see section 1.3.
[13] More information is available at https://www.thesslstore.com/blog/tls-1-3-everything-possibly-needed-know/.

with larger amount of resources and/or special purpose hardware, whilst ChaCha20 is considerably faster when run on general purpose software platforms. They can therefore both be considered for confidentiality protection of unmanned systems.

It is worth noting that when discussing confidentiality protection over radio links, the cryptographic measures should not be confused with the physical encoding of signals in radio, nor with the protocols employed for this communication. Several manufacturers use these encodings and protocols as convenient obfuscation of their transmissions, without providing cryptographic protection. A notable example of this could be a standard satellite, which often simply listens for signals with the correct encoding and protocol. It is an unfortunate misconception that simply because information is not transmitted completely openly it is secure. Multiple hacking attempts and experiments like the examples in Chapter 5 show that such encodings and protocols can be reverse engineered without substantial effort, and the problem is further emphasised by the increasing availability of Software Defined Radios (SDRs). These techniques are not intended to provide cryptographic level security, and so a radio link must be considered completely vulnerable unless recognized cryptographic measures like those discussed in this report are implemented.

4.1.1 The AES algorithm

The AES algorithm is the result of a competition held by the National Institute of Standards and Technology (NIST) in the period 1997-2000. The aim of the competition was to select an encryption algorithm to replace the old standardised encryption algorithm, DES (Data Encryption Standard). The algorithm was officially approved and released in 2001 in FIPS 197 [19].

The algorithm itself is a construction referred to as a Rijndael block cipher, made by two Belgian cryptographers. The standard was selected to have 128-bit block size and three possible key lengths – 128 bits, 192 bits and 256 bits. The most popular of these have traditionally been 128-bit, due to it being the version with best performance. Recently, the 256-bit version has seen more usage, and it is now the recommended key length moving forwards in official recommendations [20].

With modern technology we have the choice of either implementing AES in software or directly in hardware. A hardware implementation in this case consists of special instructions added onto microchips during fabrication. It is recommended that hardware implementations are used, due to the existence of side-channel timing attacks on naïve software implementations[14] [21]. Such attacks can be mitigated through careful software development, though hardware implementations are generally considered more secure and are significantly faster. Most modern microchips are produced with hardcoded AES instruction sets, which allows for hardware-based

[14] The timing attack arises as a result of the algorithm using information from the key to access array indices in a so-called substitution box or S-box. The different time taken to access different indices could then be used to help engineer the key itself. This is a weakness shared among most block cipher constructions.

AES computations, but there are still examples of AES being computed in software on systems that do not have dedicated hardware for it – especially in low-cost devices.

There are modes of AES designed to encrypt disks rather than communication – of particular note is a mode called AES-XTS [22]. Encryption of hard disks require a slightly different approach than the general communication case, due to the format in which data is stored, but AES-XTS solves this challenge and provides fast hard disk encryption.

The AES algorithm is most certainly the standard algorithm for protecting confidentiality of information on modern systems. It is implemented for almost all kinds of systems, from commercial laptops to military vehicles. Due to it being so thoroughly exposed and researched, it is considered a very safe and secure choice for confidentiality protection. Moreover, it has an inherent advantage of being commonly available on hardware instructions. This gives AES a very steep competitive edge over most of its potential competition.

4.1.2 The ChaCha20-algorithm

In a similar fashion to AES, the ChaCha20 algorithm was designed and popularised as a result of a competition. The project, called eStream, was one focusing solely on stream ciphers, and was run by a European research initiative called ECRYPT. The project ran from 2005 to 2008, with the aim to "*identify new stream ciphers suitable for widespread adoption*" [23]. The algorithm itself can be found in greater detail in its defining paper [24]. ChaCha, also referred to as ChaCha20, combined with another algorithm by the same author aimed at authentication called Poly1305[15], has since seen growing popularity and has been included in the Transport Layer Security (TLS) protocol [25], and in TLS 1.3 as the only cipher suite alternative to AES.

The overarching aim of ChaCha20's family of algorithms is to offer confidentiality protection that is designed to be simple to implement and very efficient in software. The algorithm, and its relatives, only rely on three very fundamental operations on bits: addition, bitwise XOR, and cyclical rotation of bits. These operations are implemented in hardware on all chips, as they are core to be able to perform any modern computation. This makes this family of algorithms very efficient when implemented in pure software, which in turn makes the algorithms good candidates for confidentiality protection on systems without custom hardware to accelerate encryption. Being stream ciphers, they work by generating a random-looking keystream that can be combined with the plaintext to achieve the ciphertext. The generation of the keystream is done by repeatedly applying a scrambling function. The amount of repetitions of the function can be set by parameters, and will naturally influence performance. In the proposal, the usage of 20 rounds is recommended – thus the name – but no attack better than brute force has been found for anything above 7 rounds. As such, 12 round ChaCha, referred to as ChaCha20/12, is also an option for even more lightweight applications [26] [27].

[15] The Poly1305-algorithm is a conceptually very simple message authentication code that can authenticate a message of arbitrary length, using an encryption algorithm like AES or ChaCha20. More information on the algorithm and its combination with ChaCha20 can be found in [25] [35] [36].

4.2 Integrity and authenticity protection

In most settings, hiding the content of a message is not sufficient protection of the information. Another aspect of security is that the receiver should have assurance that the message is in fact the same message that was sent from the other party, and that the message itself has not been sent by someone else. Without these assurances, the message might be practically worthless. Note that this is a completely separate issue from protecting the content, since some middle man can easily change bits in the message without knowing what they are changing from or to – and in some cases, the message can act predictably under alteration even when encrypted.[16] Most commonly, these techniques are combined with those of confidentiality, to ensure that a message is under so-called *authenticated encryption*. There are many choices for how to combine the confidentiality mechanisms from section 3.2 with corresponding solutions for integrity and authenticity. We have mostly found AES implementations paired with the Galois/Counter mode of operations we present below, in addition to the ChaCha20/Poly1305-pairing mentioned earlier.

4.2.1 Galois counter mode for AES

The AES block cipher by itself has several limitations that makes it very impractical for direct usage in communications. The most obvious one is that it only works on 128-bit blocks, which clearly is not sufficiently flexible. The Galois/Counter mode, often shorthanded to GCM, is a *mode of operation* for block ciphers such as the AES, meaning it is a wrapper algorithm that uses its given block cipher as a subroutine, applying the block cipher repeatedly in a given manner to encrypt an arbitrary length message. Unlike some other modes of operation, GCM also provides message authentication.

On the technical level, the Galois/Counter mode fundamentally changes how AES is used, transforming the cipher from a block cipher to a stream cipher. This has a side benefit of removing the timing attacks we mentioned previously on AES, as those attacks rely completely on the block cipher structure. It takes an initialization vector (IV), a secret key K, a plaintext P and some optional authenticated data (AAD). The algorithm encrypts and authenticates the plaintext, and authenticates the AAD without encrypting it. The encryption is done by encrypting a counter using AES, and using the bitwise XOR-operation on the encrypted blocks and the plaintext. The decryption algorithm takes the ciphertext, secret key, and authentication tag and either decrypts the ciphertext or return a failure message if the authentication does not check out.

Using AES in Galois/Counter mode provides an unmanned system with a communication channel that is, from a cryptographic perspective, secured against all communication attacks, as long as the cryptographic keys are handled properly on each side of the channel.

[16] Consider a message that transmit an answer to a question, either 'Yes' or 'No', as either 0 or 1 – an adversary do not know which is which, and has no means to know either. The confidentiality is thus perfectly preserved. However, said adversary could intercept the message, and flip the bit – turning a 0 into a 1, or the other way around. No confidentiality is broken, but the message is altered in a predictable fashion.

5 UAV security and vulnerabilities

There is a vast supply of different UAVs available today, both commercial and military. Some UAVs are used purely recreationally, some to take pictures such as for selling a house, or to film professionally or home movies. Others are used more industrially, like for agriculture, transport for light goods, or as entertainment in synchronized performances. Some are used by the military for military operations. This technology is proving extremely versatile and can be applied to a vast array of purposes.

With this in mind, it seems very apt that many large-scale organizations should look into including UAVs into their own operations. This chapter is intended as a limited but representative review of the different sort of solutions available on the market for such an organization from a security perspective. This will help inform further discussion about the suitability of UAVs for different applications.

In this chapter, the UAVs are divided into four general categories: recreational, commercial, enterprise, and military. These categories reflect how some UAVs generally share features and traits related to who and what they are intended for, and what measures are taken to secure them. We will expand upon this in the following sections.

5.1 Recreational UAVs

We consider a UAV to be recreational if its primary purpose is for entertainment, typically marketed towards hobbyists and other private amateurs. Many UAVs available in stores fall into this category, as they are small, cheap, and do not provide a lot of functionality – they can practically be considered as toys. An example of such a type of UAV, which has gathered quite a bit of analysis on the security front is the Parrot AR Drone 2.0 quadcopter, which we present more in detail below. Other examples include the Parrot Bebop Drone and the Ryze Tello Drone.

These products are designed to be practical and accessible, with little emphasis put on security. The UAVs will, upon activation, set up a local Wi-Fi hotspot, ready for connection from a mobile system with an appropriate application installed and running. Some come supplied with a docking station for added convenience. The UAVs have very limited flight distance, typically around a maximum distance of 100 meters, and line of sight is expected to be maintained between the operator and the UAV. This class of UAVs can be expected to have a camera connected to a HD video downlink, some local storage space for pictures and flight logs, SD card support, and GPS. Some come equipped with some more advanced manoeuvrability options such as a “return to home” function.

Figure 5.1 A Ryze Tello Drone, exemplifying recreational UAVs. The picture is by SimonWaldherr and available under Creative Commons 4.0 license.

The *AR Drone 2.0* is a quadcopter UAV manufactured by Parrot, aimed at a broad range of recreational and basic commercial applications. It is small and lightweight, weighing less than half a kilo, features a 720p camera, only 12 minutes of battery life, and is limited in operation to a maximal range of 50 meters. This range restriction follows from the usage of local Wi-Fi network communication between the UAV and the mobile device, which operates the UAV through a mobile application. The Wi-Fi connection is unencrypted. This renders the connection process completely insecure to adversaries within the operating range of the UAV.

Once a given UAV unit has been paired with a mobile device running the appropriate application, the software has the ability to enforce authentication based on MAC addresses (referring to Media Access Control addresses, not Message Authentication Codes) that allow only connections identified as originating from the correct MAC address. However, such MAC address-based schemes are known to be easy to overcome through spoofing. Seeing as the handshake is in plaintext, an adversary can simply observe an accepted connection and extract the MAC address from it. Thus, this connection boot-up must be considered completely insecure, and can only be done under very controlled circumstances where no adversaries might be eavesdropping on wireless traffic.

The UAV limits the number of connections on the network to a single user, in an attempt to protect against adversaries. However, current Wi-Fi networks are known to be vulnerable to so-called deauthentication attacks [28], where an adversary can send unauthenticated packages to the router (in this case, the UAV) and activate a general deauthentication of all connections and requiring that a new connection be established. This is by itself a general DoS-attack, which, in the context of UAVs, completely inhibit usability.

Moving onto the link itself, there are many critical vulnerabilities. In a paper addressing the security of the AR Drone 2.0 [28], the authors discover several such vulnerabilities. Most crucially, both the telnet and FTP connections were found to be accessible wirelessly and devoid of any authentication mechanisms (including the unencrypted Wi-Fi). As such, an adversary within range can freely obtain wireless access to the file system, enabling any software attack such as reading files, injecting commands, or injecting custom software/firmware. This crucial vulnerability renders the UAV permanently insecure to adversaries within wireless range.

5.2 Commercial UAVs

Similar to recreational UAVs, commercial UAVs are not systems intended to meet high demands on reliability and features. Yet, they are considerably more advanced than recreational systems. This class consists of accessible general-purpose UAVs that use radio communication to facilitate much higher operational distance than pure Wi-Fi-based connections – typically in the range of a few kilometres. The UAVs are overall small, ranging between 10 and 30 cm across, and contains radio communication, UHD camera, GPS, and advanced flight features. It is worth noting that although in many cases there is not a direct connection between a mobile device and the UAV, there are often configurations where the UAV is operated by a mobile device through a remote control gateway unit that communicate over both Wi-Fi and radio link.

We discussed some of the potential vulnerabilities associated with Wi-Fi in section 5.1. In addition, it is important to keep in mind that as long as a mobile device is kept in the loop, the system as a whole has a much larger attack surface, due to the high number of attack vectors for mobile devices (as presented in for example [29]).

Figure 5.2 The DJI Phantom 3 Standard is a well-studied commercial UAV using a remote controller with a mobile device for control. The picture is taken by Marco Verch and is available under Creative Commons License 2.0.

The *DJI Spark* was released in 2017. It costs NOK 3000-4000, and its battery lasts only 16 minutes of flight. The update files are signed with RSA-SHA256, and some parts of the files are encrypted with AES[17]. The UAV uses a 2.4GHz WPA-2 protected Wi-Fi hotspot that is restricted to a single connection only to communicate with a mobile device with the DJI GO 4 application, or another 2.4GHz wireless link to connect to a remote controller. The UAV, like other DJI UAVs, communicate with the DJI server. This communication does not seem to require any authentication, but the communication is encrypted, using AES in CBC mode[18] with hardcoded 256-bit keys, one for requests and one for responses. Some analysis let these keys be extracted.This allowed an attacker to inject a package to change the Wi-Fi password and de-authorise the user, effectively gaining full control. The attack was claimed to work on "*all the Wi-Fi manageable UAVs that are compatible with the DJI Assistant 2 applications*", which included most UAVs made by DJI[19].

The Phantom series is a series of UAVs produced by DJI. The *Phantom 3* is a bigger UAV than those discussed in Section 5.1, at 1.2 kg weight and 25 cm diagonal size. It features a 2K camera. The range is more impressive than that of a pure Wi-Fi-based UAV, claiming a range of

[17] The AES keys were leaked on GitHub by a DJI developer, and enthusiasts made a tool that allows for decryption of these files. News story for example at https://www.theregister.co.uk/2018/01/25/dji_github_public_repo_crypto_key_foolishness
[18] CBC, or cipher block chaining, is a mode of operation for block ciphers where each plaintext is added through XOR with the previous ciphertext before it is encrypted. This adds additional security as compared to encrypting each block separately, but does not offer for instance integrity the way GCM mode does.
[19] This attack is detailed at the blog post https://embedi.org/blog/dji-spark-hijacking.

up to 1000m due to a 5.725-5.825 GHz radio remote controller. The remote acts as a Wi-Fi access point, relaying control and command messages from a mobile device, with a 2.400-2.483 GHz Wi-Fi network between the remote controller and the mobile device. It comes with the DJI GO Application, to be used for controlling the UAV.

The Phantom 3 UAV is studied extensively in [8] and [14]. The UAV uses a proprietary protocol to communicate between the remote controller and the UAV, as well as for encrypting its file system. It does however keep track of flight logs through a custom file encoding they name DAT files, stored on an internal SD card. The content of these DAT files is obfuscated in a fashion similar to encryption; however, the algorithm has been determined to be extremely weak and easily broken[20]. There are also some similar text files stored on the mobile device containing logs, and these files are protected by a much stronger encryption scheme.

In contrast to UAVs such as the AR Drone 2.0, the DJI Phantom 3 only allows a single Wi-Fi connection. This helps mitigate several of the Wi-Fi attacks that the AR Drone 2.0 has been shown to be vulnerable to. One such attack was named *SkyJack*, and consisted of software that, when running on an adversarial drone, enabled the adversarial drone to hijack any AR Drone 2.0 it came in proximity with[21]. Similarly, [14] found that they were not able to connect through telnet or use telnet as a basis for a denial of service attack. As such, some attack vectors have been covered.

There are however still several vulnerabilities remaining. Like most UAVs, the Phantom 3 Standard is vulnerable to GPS attacks like spoofing and jamming. More severely, [14] found that the camera allowed FTP connection attempts, and ,by utilizing reverse engineering and decompilation of the DJI GO application, were able to obtain the root FTP password, giving full read/write access to media devices. The decompilation also revealed a configuration file containing no-fly zones, which they could then modify and recompile, establishing a software integrity attack. An FTP connection was established with the UAV using the password, giving full access to the entire file system including (encrypted) firmware, which enabled adding new files to the root, like initialisation files that automatically run on boot. The controller could also be accessed through FTP with the same password.

The controller uses radio signals to send flight instructions to the UAV. A frequency spectral analysis of the signals could be used to reveal the signals used for each flight instruction, allowing for the spoofing of commands and options for replay attacks depending on the encryption on the radio communication. This attack has been said to be promising [14].

There is little about the encryption of Phantom 4. In the update logs available from their website, DJI note at one point that they "*improved transmission encryption*", implying that such encryption exists – but what it is not clear.

[20] The algorithm consisted of using the tick number of the packet mod 256 as a key, and XOR each byte with this key. This is an extremely weak algorithm, and it is not clear why such a weak encryption algorithm is used to protect the DAT files (though the files themselves can be viewed in some circumstances as more or less harmless).
[21] Much information can be found on the author's own webpage at https://samy.pl/skyjack/.

For the old Phantom 2, the extender IP was accessible through SSH and it was pubic knowledge that the username was root and password 19881209.

5.3 Enterprise UAVs

Even for mainstream suppliers, some products are not intended for sale over counter. We consider a product to be *enterprise* if it is primarily manufactured for larger organizations, such as industrial corporations or governmental entities. This can often result in a much more robust, high quality, specialized product that is customizable to the needs of the customer. Overall, we find that these systems advertise more security features than the recreational and commercial systems, especially when marketed towards applications within military and law enforcement. In spite of this, products still appear to have critical vulnerabilities, many of which it shares with the higher end commercial systems.

One critical part of these UAVs which retain many of the vulnerabilities of cheaper options is the Wi-Fi. Many UAVs even at a professional or enterprise level include a mobile device with some application in the operating configuration. In one example given in a presentation[22] at the RSA Conference 2016 and Black Hat 2016, a UAV intended for use in law enforcement was examined. The operation configuration consisted of a mobile device with flight planning software, connected to a remote control and telemetry box through 802.11 Wi-Fi with standard Wi-Fi encryption. This was in turn connected to the UAV through a 2.4GHz short range Wi-Fi link as well as an XBee 868LP radio link with a 2km range (and a video link going back).

As we have outlined previously, this setup is not secure. When discussing the Parrot AR Drone 2.0 in Section 5.1, we noted that the Wi-Fi connection between the mobile device and the Remote Control has several vulnerabilities similar to that of the AR Drone 2.0. It can for instance be subjected to a deauthorisation attack, and an adversary can re-establish the link using their own device. Such a connection also gives access to encryption keys for the UAV. The most limiting part of this attack is simply the range, like with the AR Drone 2.0.

Perhaps more interestingly, the presentation also details how one might go about performing a man-in-the-middle attack on the telemetry link. By use of some surprisingly powerful features of the chip, such as the support of API commands with no authentication required, an adversary can instruct the chips on both the Remote Control and the UAV itself to redirect traffic through the adversarial device. Through reverse engineering the software on the Android application, the adversary can obtain commands that they are free to send. These commands have no integrity or authentication protection whatsoever. It was observed in the presentation that there existed an onboard encryption feature for the XBee chips, but the manufacturer had chosen not to enable the feature.

[22] The presentation is called *Hacking a Professional Drone* by Nils Rodday, held at RSA Conference 2016. The presentation can be found at https://www.rsaconference.com/writable/presentations/file_upload/ht-w03-hacking_a_professional_police_drone.pdf.

Figure 5.3 Enterprise UAVs are designed for industrial applications and similar operations, for example for usage in firefighting or agriculture. The picture is licenced for free commercial usage.

There are several UAVs in the Matrice-series that fit this grade. The *Matrice 600* measures over one and a half metre in length, weighing around 10 kg. It is able to handle a payload up to 5 kg. The *Matrice 200* is a little over half a metre, weighing 6 kg with a payload up to 1.45 kg. Both use a remote controller, with two frequencies: a radio band at 5.725-5.825 GHz and Wi-Fi at 2.4 GHz (Same as the Phantom 3). Like the Phantom 3, it is used with a mobile device. The range is said to be around 8 km from the transmitter to the UAV. The newest products in the Matrice series, called Matrice V2, use a transmission system developed by DJI called OcuSynch 2.0 whose data is "*fully encrypted using AES-256*" according to DJI's own website.

The Matrice models support DJIs Onboard SDK. In its documentation, there is detailed information about their OPEN protocol. The protocol includes a 3-bit field called ENC, which when set to 0 indicates "no encryption" and 1 indicates "AES encryption". They write the encryption is in the interface between the Onboard Embedded System (OES) and the autopilot, onboard the UAV. They intend for this to be used if data is transmitted wirelessly between components. The encryption only protects the data in the frame. By default, this encryption is turned off.

The Matrice series has recieved more attention from hackers than many other enterprise products. One of the reasons might simply be because it is produced by DJI, which seems to be a common target for many researchers and enthusiasts. There are multiple articles[23] detailing users and companies managing to reverse engineer and modify software applicable to the entire DJI product line, which includes its enterprise products. This includes no flight zones, and software limitations on features such as speed and altitude. One company, called CopterSafe, is mentioned in the articles as being able to remove such restrictions for the Inspire 2 and Matrice

[23] VICE has the 2017 article "*DJI Is Locking Down Its Drones Against a Growing Army of DIY Hackers*" at https://www.vice.com/en_us/article/3knkgn/dji-is-locking-down-its-drones-against-a-growing-army-of-diy-hackers, and another article detailing techniques from 2017 is "*Here's How To Hack Your DJI Drone*" at https://www.wetalkuav.com/coptersafe.

600 products, as well as adding support for a third party thermal camera with HD downlink. The camera can also be installed on a Phantom 3. As these are considered to be minor entities and amateurs, this heavily indicates that the software is not secure against an advanced adversary.

The *Mavic 2 Enterprise* uses OcuSynch 2.0 with 8 km transmission range. The DJI Mavic 2 Enterprise has "*local encryption*": "*when password protection is enabled, users are required to enter their password each time they activate the UAV, link the remote controller with the UAV, and access the UAV's onboard storage*". One can make similar comments to Matrice regarding the interest of hackers, and some stories mentioned Mavic 2 Pro in particular[24] when discussing how software can be modified.

The *GRIFF Guardian* is one of several models made by Norwegian UAV producer GRIFF. The UAV is intended for use in law enforcement. It operates using a remote control station and a telemetry link over several bands, where much effort has been spent on safety precautions so that the link stays up. Nothing is published about their security solutions.

5.4 Military UAVs

Outside of the standard market, there is a very large marked for UAVs specifically designed for military purposes. These are produced by a few actors, sometimes through governmental programmes. Typically, such products spout a long development cycle and at very high costs, in part due to high demands for customizability and secrecy. They often go through several significant certification processes including many different testing procedures before they are considered cleared for usage. Common for almost all of these products are a lack of information about their security solutions, in particular cryptographic mechanisms.

The following is a short overview of some prominent actors and their military systems. In addition to the systems themselves, we note what sort of security solutions are advertised or known publically.

5.4.1 Lockheed Martin

Lockheed Martin has a large stake in the market of military unmanned systems, with several products over many applications. Their most prominent product is perhaps the large stealth-capable fixed-wing *UAV RQ-170 Sentinel*, which has seen operational use in Afghanistan. Such a UAV was captured through a reported hijack by the Iranian government while flying over Iranian airspace in December 2011[25]. It was claimed by Iranian engineers that the hijack happened through a GPS spoofing attack, coupled with jamming of control signals. This was however disputed by American engineers, who claimed such an attack would not work on the RQ-170. After its capture, it was thoroughly investigated and attempted reverse engineered -

[24] For example the article "*Hack your Mavic 2 Pro to Acquire Attitude Mode*" from 2019, which can be found at https://dronedj.com/2019/05/25/hack-your-mavic-2-pro-to-aquire-attitude-mode.

[25] One news source on this is the article "*Iran allegedly used GPS exploit to capture US drone*", published by The Verge and available at https://theverge.com/2011/12/16/2640663/iran-gps-exploit-capture-us-drone.

thus, making it a clear real world example of a system being attacked through an advanced adversary as defined in chapter 2. Iranian officials have claimed that data was extracted and "fully decoded" from the system, though United States officials again disputed this due to their security measures on the RQ-170 and instead suggest the UAV was lost due to a technical malfunction.

There are several other unmanned systems produced by Lockheed Martin as well. One is the *Desert Hawk*, used for reconnaissance and surveillance by for example the British Army. Lockheed Martin states in their product brochure that the Desert Hawk has "*commercial and military grade communication options*". It is unclear and unspecified what such options could be, though it could indicate an AES-256 implementation. A FlightGlobal article[26] concerning the Desert Hawk III states that it "*allows better data encryption and data transmission via 3G and 4G mobile telecommunication networks*", though no further illumination on or source for this claim was provided. Lockheed Martin also has AUVs such as the *EMETT* and *Submatt*, but make no mention of encryption or data security measures concerning these systems. They also have smaller UAVs such as the Indago UAV, also with no mention of encryption. They do mention encryption concerning their mobile application platform, stating that their application "*meets mobile security requirements including FIPS 140-2 compliance*" and "*establishes a secure, encrypted connection and authenticates users for military-grade security*".

5.4.2 AeroVironment

AeroVironment is another large-scale actor in the field of unmanned systems in the United States. Their *Raven* model is described by AeroVironment on their website as *"a lightweight Unmanned Aircraft System (UAV) designed for rapid deployment and high mobility for both military and commercial applications*", and they claim that it "*is the most prolific small UAS deployed with the U.S. Armed Forces*". Airforce Technology writes that it "*offers higher communication security through signal encryption*"[27], and a United States Army brochure hosted on the Federation of American Scientists state that it "*incorporates secure Global Positioning System navigation*" and that "*the digital data link incorporates encryption*"[28]. In addition to the Raven, AeroVironment has a product called *Puma*, which they explicitly state uses AES-256 encryption, and one called *Wasp AE*, where no mention is made of cryptography.

[26] The article is called «*AUVSI: Desert Hawk gains endurance, updated software and systems*», is published on FlightGlobal and can be found at https://www.flightglobal.com/news/articles/auvsi-desert-hawk-gains-endurance-updated-software-398761/.
[27] This is found at https://www.airforce-technology.com/projects/rq11braven/.
[28] The brochure is available online at https://fas.org/man/dod-101/sys/land/wsh2013/278.pdf.

Figure 5.4 A MQ-9 Reaper flying for the US Army, produced by General Atomics. The picture is provided by the U.S. Air Force and released to the public domain.

5.4.3 Boeing

Boeing has several medium-sized UAVs for military purposes, mainly through their subsidiary Insitu Inc. Their *ScanEagle* is a "*long-endurance unmanned aerial system (UAS)*", with "*intelligence, surveillance and reconnaissance services*". Boeing writes that the ScanEagle has a C2 (command and control) datalink that can operate either encrypted or unencrypted, that it has a video datalink that can be "*either analog or digital encrypted*". Industry Daily reported in 2019 that tests on several Insitu products were done with a digital data link called "Bandit", which reportedly is "AES capable".

5.4.4 Others

In contrast to the other products discussed in this category, the Black Hornet is a much more small form factor UAV. Its security features are not openly disclosed more than that it uses AES-256.

There are many other notable actors in the field, such as Textron, Northrop Grumman, and General Atomics Aeronautical. We have however not found much on their products.

6 Conclusion

Unmanned systems have become progressively more common in use over the past decade. Whilst they provide much functionality, there are still many issues to consider regarding their security, especially for critical operations like industrial or military applications. In this report we have looked at many types of, and examples of, attacks on unmanned systems, as well as various ways they are addressed and could be addressed by cryptographic mechanisms in unmanned aerial vehicles. We established much of the required language to discuss this in Chapter 2.

We categorized and examined the different types of attacks on unmanned systems in Chapter 3. We divided the attacks into four domains: hardware, side channels and fault injection, software, and communication. We then described how adversaries might exploit facets of these domains to compromise confidentiality and integrity of information. We highlighted that there are many attack vectors on unmanned systems, and any of these domains or a combination of them could be used to attack any given system. We also highlighted that it is important to consider the context within which a given unmanned system will operate in order to accurately assess the potential risks of attacks. Finally, we briefly outlined some additional attack vectors that we did not cover in-depth, since they cannot be addressed with cryptographic mechanisms.

We then examined the specific cryptographic mechanisms that was most prominent among commercial unmanned aerial vehicles in chapter 4. We found that the Advanced Encryption Algorithm (AES) was by far the most common mechanism to ensure confidentiality, although some smaller systems have recently started favouring ChaCha20 due to its efficiency in software. The AES algorithm as originally designed protects communication, and can also be used in a variant to protect hard drives. The AES algorithm can be used in a mode of operations called Galois/Counter mode, which ensures authenticity of the information in addition to the confidentiality. Together, this authenticated encryption scheme was by far the most implemented and referenced.

In chapter 5, we reviewed efforts to bypass the security of different unmanned aerial vehicles. We divided the vehicles into four types: recreational, commercial, enterprise, and military. The division was done based on their general audience and expected operational environment. We found that most of the unmanned systems in question implemented some form of cryptographic mechanisms, but the mechanisms were often either lacking or not included in the platform in an overall secure way. Often, many of the attacks we listed in Chapter 3 had been successful at compromising these goals. In particular, reverse engineering and injection of software was very prominent, and none of the studied systems resisted prolonged attacks where the forensic analysts had physical access.

When breaking down the unmanned aerial vehicles into categories based on intended environment, there were clear similarities between systems within each category. Recreational vehicles were generally very badly secured if at all, relying entirely on scantly defended Wi-Fi networks to be controlled. Commercial vehicles and enterprise vehicles had potential vulnerabilities due to their reliance on mobile devices and Wi-Fi connections between those devices and remote controllers, and were badly secured against reverse engineering with sensitive information such as passwords readable from the extracted software. Military systems are mostly not openly available, but there have been examples of military systems lacking encryption on communication and being vulnerable to GPS-based attacks. When information were available, it seems overwhelmingly like AES-256 in Galois Counter Mode is the industry standard and widely considered “military grade encryption”, providing authenticated encryption on communication links.

From the body of literature we have surveyed, it seems clear that there is an enormous attack surface against unmanned systems and not enough security features have been implemented by developers. This is particularly important since cryptographic mechanisms which provide protection against many of the listed attacks already exist today and are implemented in some systems, but not others. Future work should look at which security requirements should be placed on unmanned systems to decide whether they are at all suitable for critical operations, as well as investigating ways in which existing cryptographic mechanisms could be introduced to commercial off-the-shelf products to provide more satisfactory security.

References

[1] F. Mancini, S. Bruvoll, R. Fardal, J. H. Wiik, B. M. Greve, L. E. Olsen og B. Bjerketveit, «Information security for unmanned and autonomous vehicles - main challenges and relevant operational concepts,» Norwegian Defence Research Department (FFI), 2019.

[2] E. M. a. J. A. H. Huang, «Autonomy Levels for Unmanned systems (ALFUS) Framework,» NIST Special Publication 1011-II-1.0, 2007.

[3] H. Z. John D Day, «The OSI reference model,» *Proceedings of the IEEE,* pp. 1334 - 1340, January 1984.

[4] Nasjonal Sikkerhetsmyndighet, «Temarapport kvanteresistent krypto,» 2017.

[5] Lily Chen et al, «Report on post-quantum cryptography,» US Depertment of Commerce, National Institute of Standards and Technology, 2016.

[6] N. Rodday, «Hacking a Professional Drone,» i *RSA Conference 2016*, San Francisco, 2016.

[7] F. K. F. I. A. M. Hana Bouafif, «Drone Forensics: Challenges and New Insights,» i *9th IFIP INternational Conference on New Technologies, Mobility and Security (NTMS)*, 2018.

[8] C. M. I. B. F. B. Devon R. Clark, «DROP (DRone Open source Parser) your drone: Forensic analysis of the DJI Phantom III,» *DFRWS 2017 USA - Proceedings of the Seventeenth Annual DFRWS USA,* 2017.

[9] E. Dubrova, «Anti-Tamper Techniques,» Royal Institute of Technology, Stockholm, Sweden, 2015.

[10] J.-M. S. I. V. Dusko Karaklajic, «Hardware Designer's Guide to Fault Attacks,» *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS,* 12 December 2013.

[11] V. M. T. K. S. M. Raphael Spreitzer, «Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices,» *IEEE COMMUNICATIONS SURVEYS & TUTORIALS,* 2017.

[12] F.-X. S. Marcel Medwed, «Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices,» i *Progress in Cryptology - AFRICACRYPT 2010*, South Africa, 2010.

[13] M. C. L. A. A. G. L. Ablon, «Markets for Cybercrime Tools and Stolen Data: Hackers' Bazaar,» Rand Corporation, 2014.

[14] B. C. G. B. R. R. Fernando Trujano, «Security Analysis of DJI Phantom 3 Standard,» 2016.

[15] R. C. Johann Pleban, «Hacking and securing the AR.Drone 2.0 quadcopter - Investigations for improving the security of a toy,» i *Proceedings of SPIE - The International Society for Optical Engineering*, 2014.

[16] D. Kovar, *Forensic Analysis of sUAS aka Drones*, SANS Digital Forensics, 2015.

[17] S. Garfinkel, «Anti-forensics: Techniques, detection and countermeasures,» *2nd International Conference on i-Warfare and Security*, vol. 20087, pp. 77-84, 2007.

[18] J. A. B. T. E. H. Daniel P. Shepard, «Drone hack: Spoofing attack demonstration on a civilian unmanned aerial vehicle,» 2012.

[19] P. NIST FIPS, «197: Advanced encryption standard (AES),» Federal information standards publication, 2001.

[20] A. Daniel, B. Lejla og e. al, «Initial recommendations for long-term secure post-quantum systems,» PGCRYPTO. EU. Horizon 220, 2015.

[21] D. J. Bernstein, «Cache-timing attacks on AES,» 2005.

[22] L. Martin, «XTS: A mode for AES for encrypting hard disks,» *IEEE Security & Privacy*, vol. 8, nr. 3, pp. 68-69, 2010.

[23] ECRYPT, «Call for Stream Cipher Primitives,» 12 April 2005. [Internett]. Available: www.ecrypt.eu.org/stream/call/. [Funnet 28 March 2019].

[24] D. J. Bernstein, «ChaCha, a variant of Salsa20,» 2008.

[25] e. a. Langley, «RFC7905: ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS),» Internet Engineering Task Force (IETF), 2016.

[26] J.-P. Aumasson, S. Fischer, S. Khazaei, W. Meier og C. Rechberger, «New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba,» 2008.

[27] G. Procter, «A Security Analysis of the Composition of ChaCha20 and Poly1305,» *IACR Cryptology Cryptology ePrint Archive,* 2014.

[28] J. Milliken, V. Selis, M. Yap og A. Marshall, «Impact of Metric Selection on Wireless DeAuthentication DoS Attack Performance,» *IEEE Wireless Communications Letters,* vol. 2, pp. 571-574, 2013.

[29] R. C. J.S. Pleban, «Hacking and securing the AR- Drone 2.0 quadcopter: Investigations for improving the security of a toy,» *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications,* vol. 9030, 2014.

[30] F. M. D. S. M. La Polla, «A survey on security for mobile devices,» *IEEE communications surveys & tutorials,* vol. 15, nr. 1, pp. 446-471, 2012.

[31] J.-P. a. F. S. a. K. S. a. M. W. a. R. C. Aumasson, «New Features of Latin dances: analysis of Salsa, ChaCha, and Rumba,» 2008.

[32] P. FIPS, «46-3. Data Encryption Standard (DES),» National Institute of Standards and Technology, 1999.

[33] P. NIST FIPS, «46-3. Data encryption standard,» Federal Information Processing Standards. National Bureau of Standards. US Department of Commerce, 1977.

[34] D. J. Bernstein, «The Salsa20 family of stream ciphers,» *Springer*, 2008.

[35] R. N. Akram, P.-F. Bonnefoi, S. Chaumette, K. Markantonakis og D. Sauveron, «Secure Autonomous AVs Fleets by Using New Specific Embedded Secure Elements,» IEEE, 2016.

[36] H. Huang, E. Messina og J. Albus, «Autonomy Levels for Unmanned systems (ALFUS) Framework,» NIST Special Publication 1011-II-1.0, 2007.

[37] D. P. S. J. A. B. T. E. H. Andrew J. Kerns, «Unmanned Aircraft Capture and Control via GPS Spoofing,» *Journal of Field Robotics*, vol. 31, 2014.

[38] D. J. Bernstein, «The Poly1305-AES message authentication code,» i *International Workshop on Fast Software Encryption*, Springer, 2005, pp. 32-49.

About FFI

The Norwegian Defence Research Establishment (FFI) was founded 11th of April 1946. It is organised as an administrative agency subordinate to the Ministry of Defence.

FFI's MISSION

FFI is the prime institution responsible for defence related research in Norway. Its principal mission is to carry out research and development to meet the requirements of the Armed Forces. FFI has the role of chief adviser to the political and military leadership. In particular, the institute shall focus on aspects of the development in science and technology that can influence our security policy or defence planning.

FFI's VISION

FFI turns knowledge and ideas into an efficient defence.

FFI's CHARACTERISTICS

Creative, daring, broad-minded and responsible.

Om FFI

Forsvarets forskningsinstitutt ble etablert 11. april 1946. Instituttet er organisert som et forvaltningsorgan med særskilte fullmakter underlagt Forsvarsdepartementet.

FFIs FORMÅL

Forsvarets forskningsinstitutt er Forsvarets sentrale forskningsinstitusjon og har som formål å drive forskning og utvikling for Forsvarets behov. Videre er FFI rådgiver overfor Forsvarets strategiske ledelse. Spesielt skal instituttet følge opp trekk ved vitenskapelig og militærteknisk utvikling som kan påvirke forutsetningene for sikkerhetspolitikken eller forsvarsplanleggingen.

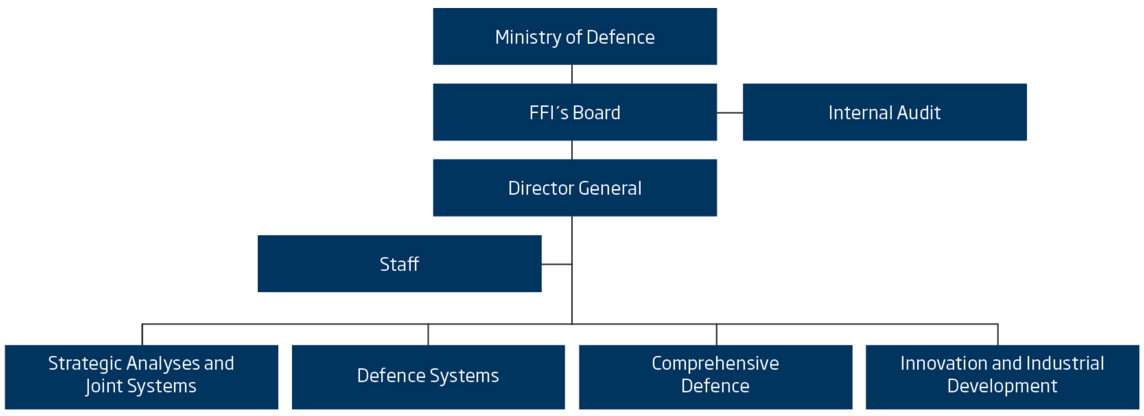
FFIs VISJON

FFI gjør kunnskap og ideer til et effektivt forsvar.

FFIs VERDIER

Skapende, drivende, vidsynt og ansvarlig.

FFI's organisation


Ministry of Defence
FFI´s Board
Internal Audit
Director General
Staff
Strategic Analyses and Joint Systems
Defence Systems
Comprehensive Defence
Innovation and Industrial Development