Master's Thesis

Authentication and Integrity for ADS-B: Design and Implementation of a Security Model for Air Traffic Control Signals

by

Markus Fögen

December 07, 2018

Technische Universität Kaiserslautern
Department of Computer Science
Distributed Computer Systems Lab

Examiner: Prof. Dr.-Ing. Jens B. Schmitt
Supervisor: M. Sc. Carolina Nogueira

Abstract

In a short time from now, the *Automatic Dependent Surveillance - Broadcast* (ADS-B) standard will become the new mandatory air traffic control system all over the world. While bringing a lot of advantages for pilots and air traffic controllers to increase the safety in the air, the standard comes with a complete lack of basic security mechanisms. This thesis presents a design proposal for enabling the possibility to verify the integrity and the authentication of ADS-B messages. Using a second-channel communication, a security model will be established that is still completely compatible with the existing system. A proof of concept is implemented to research parameter values needed to establish a secure and reliable transmission of the ADS-B signals. By presenting the different possibilities in the system design and by discussing a set of possible attacker models, the reliability of the model will be shown.

In naher Zukunft soll der *Automatic Dependent Surveillance - Broadcast* (ADS-B) Standard traditionelle Radar-Systeme zur Luftraumüberwachung weltweit ersetzen. Für die Koordination von Flugzeugbewegungen und Kollisionsvermeidungen stellt ADS-B einen großen Fortschritt dar. Doch bei der Entwicklung des Standards wurden essentielle Sicherheitsmaßnahmen vernachlässigt. Diese Arbeit präsentiert einen Vorschlag für ein Sicherheitsmodell, welches es erlaubt, die Integrität von ADS-B Nachrichten sicherzustellen und den Autor der Nachrichten zu verifizieren. Über einen zweiten Kanal werden Sicherheitsmechanismen zusammen mit den ADS-B Daten übertragen, ohne das bisher bestehende System zu verändern oder zu stören. Mithilfe einer Beispielimplementierung werden die notwendigen Parameter für eine sichere und verlässliche Übertragung erforscht. Die Zuverlässigkeit des vorgeschlagenen Systems wird mithilfe unterschiedlicher Designvorschläge und über Angriffsmodelle diskutiert.

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Masterarbeit mit dem Thema **"Authentication and Integrity for ADS-B: Design and Implementation of a Security Model for Air Traffic Control Signals"** selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

(Ort, Datum) (Unterschrift)

Contents

1. Introduction

This thesis presents a proposal to add missing security features as hidden information into the transmission of Air Traffic Control (ATC) - ADS-B - messages. Additional information shall be hidden in the modulation of the digital signal, invisible to systems not searching for it. This way, full backward compatibility to the existing system is ensured while offering security features to every interested party. A combination of symmetrical cryptography (Message Authentication Codes - MAC) and asymmetrical cryptography (digital signatures) shall be used to provide accurate knowledge about the sender of an ADS-B Message (authentication) as well as ensuring the integrity of all messages. While designing the additional measures, the computational effort of the offered proposals is still kept in mind—with regards to the often limited computational power of the target group.

Modern *Air Traffic Control*-Systems are based on a combination of two radar systems: The primary surveillance radar (PSR) and the secondary surveillance radar (SSR). The latter system, which is crucial for not only observing, but also identifying aircraft in the observed airspace, is currently subject of a renewal process. Automatic Dependent Surveillance-Broadcast (ADS-B) is chosen to become the mandatory standard for air traffic surveillance by 2020 in the United States of America and the European Union [54]. The main principle of ADS-B is the regular, self-reliant broadcast of information like the current position, altitude or identification of the aircraft or ground stations via *ADS-B out*. Due to its open nature, since no kind of encryption is used to hide the information, ADS-B Data can be received by every interested party using quite low-priced equipment (compared to traditional ATC) [56].

ADS-B comes with a lot of advantages and features for air traffic control and pilots. For pilots, aircraft equipped with *ADS-B in* profit from a greatly increased situational awareness, since they are able to observe the surrounding airspace independently from ground control stations or country- or vendor-specific traffic alert and collision avoidance systems (TCAS). Ground stations profit from the very up-to-date overview over all aircraft in range, since position reports are sent out every 400 to 600 ms. In contrast, the update rate of traditional SSR depends on the rotation rate of the antenna. Additionally, position reports sent via ADS-B are based on global navigation satellite systems (GNNS) and therefore in general they are very precise. This increased precision and timeliness can be used to tighten the separation standards, allowing a more efficient usage of the airspace [30, 54]. Additionally, the usage of ADS-B allows air traffic controllers to keep track of regions, that are hard or even impossible to observe via traditional radar systems. As an example, Nav Canada already started using ADS-B in 2009 to provide coverage in the Hudson Bay Area, claiming the resulting reduced separation-standard from 80 nmi to just 5 nmi "could cut airline fuel costs by about C$10 million [...] a year" [54, 58].

On the downside, due to its open nature, ADS-B is a quite easy target for attackers. Since messages are neither encrypted nor authenticated, the usage of ADS-B offers a lot of possibilities to interfere with the system. Attacks range from simple spoofing up to complex assaults like Denial of Service (DoS) attacks, that could render the entire ATC of an airport useless when relying fully on ADS-B. Early on, researchers and hackers presented a set of theoretical and practical attacks that are possible even with low cost hardware based on software defined radios (SDR) and open libraries [15, 56].

The missing encryption of ADS-B messages, combined with the always included call sign of the aircraft, makes aircraft very easily traceable. Privacy concerns are raised by some researchers at this point [15]. Every moving aircraft, even private ones, that are obligated to use ADS-B, may be tracked and even data-mined fairly easy. On the other hand, this open nature may also be the greatest advantage of the system. Due to the low requirements, every country, airport and other aircraft may profit from the benefits in ATC. The described increased situational awareness for pilots would no longer be possible without a complex and potentially infeasible key management system.

The greatest concern may be the missing authentication feature, making identity fraud easy. To spoof the message or even impersonate another flight, the attacker could simply alter the call sign sent out by their machine. Combined with a spoofed position report, a potential malicious attacker could hide forbidden actions up to terrorist activities. While a spoofed position could be detected by additional sanity checks and multilateration systems [64], an identity fraud can not be recognized without a proper authentication check. Likewise, message manipulation, for example by corrupting the message at the receiver's end, can not be detected without a reliable integrity verification.

Contributions and Outline

To face the challenges described and to allow the operation of ADS-B in a safe and secure environment, the contributions of this work are as follows:

- The theoretical and practical background of Air Traffic Surveillance using SSR Mode S and ADS-B is provided to explain the need for a security model and the foundation the security model is based on. Basic explanations of used concepts and algorithms that are used for bringing security to ADS-B are elucidated.

- A literature research is presented to discuss the possibilities to establish a security model and to give an overview over recent research that was provided to face the challenges ADS-B is coming with

- A design of a security model is presented, that will ensure a secure and reliable transmission of ADS-B messages to guarantee the safety of air traffic control. Parameters are determined to allow the verification of the integrity and the authentication of messages without giving adversaries the possibility to fraud realistic messages.

- Using a proof of concept, the requirements of a secure transmission is studied further and a foundation for a reasonable parameter choice is set.
- Several attacker models are discussed in order to gain knowledge about the security of the proposed model.

The remainder of this work is organized as follows: The next chapter will provide background information regarding air traffic control, ADS-B, the OpenSky Network and digital modulation used in ADS-B. Basic cryptography and coding theory concepts and algorithms used in this thesis are explained. Chapter 3 will present a literature research to study the needs, requirements and possibilities on how to protect the modern air traffic control messages. Chapter 4 will give a short overview of the possible measures to inject additional information into an ADS-B message. The design of the security model itself is explained in Chapter 5. Step-by-step, a solution is presented on how to add additional information into the existing system to provide integrity and authentication verification. Chapter 6 discusses a proof of concept implementation and experiments made to determine important security parameters and design decisions. The choices for the parameters used and alternatives are discussed in Chapter 7. Presenting a set of attacker models, the reliability of the proposal is shown. In the end, Chapter 8 concludes the thesis and gives an overview over future work arising from the presented design.

2. Background

2.1. Historical Background

When thinking of Air Traffic Control-Systems, most people usually think about the so called "Primary Surveillance Radar" (PSR). Via a rotating antenna, that emits radio signals to the air and captures reflections, this kind of radar can determine the position of aircraft, birds, weather- and land-features. These positions are mainly calculated based on the bearing of the antenna while receiving the reflection and the signals time of flight. The first airport to introduce an air traffic control system based on PSR was the Croydon Airport in London in 1921 [67]. While giving the possibility to determine (roughly) the position of the targets, an ATC based only on PSR offers no way to identify them.

The strong need to identify the observed aircraft rose up in World War II. Especially the British military airspace surveillance stations were in a need of a more advanced surveillance system to keep the own and the enemy aircraft apart to protect returning planes from German persecutors. This led to the invention of the so called "Identification Friend or Foe" system. In civil use, this system became later known as "Air traffic control radar beacon system (ATCRBS)". Similar to the PSR, a so called "Secondary Surveillance Radar (SSR)" emits radio signals (*interrogations*) from a ground station in the direction of the aircraft. But now, the aircraft carries a transponder, which replies to these interrogations by sending out *responses* [69]. Using this system, the ground station is able to easily poll information like the height of flight (altitude), velocity and the identification of the aircraft. On the downside, using SSR there is now a need for co-operation by the aircraft, while the PSR is able to determine position of every aircraft within its coverage on its own (as long as the aircraft is able to reflect radio waves) [61]. What type of information is polled is determined by the type of interrogation sent by using a system of *modes*. As an example, an interrogation of Mode C is used to request an aircraft's altitude. While the SSR solves the problem of identifying aircraft in range and simplifies the exchange of some information (since ground control can easily poll them instead of communicating with the pilot personally), it still inherits some of the problems of the PSR. Due to the low resolution of the radar, position reports are still very vague. Caused by the rotation rate of the antenna (usually about 12s/rotation) [54], the current observation is outdated most of the times, forcing ATC to preserve a high safety distance between aircraft. Still, objects can cover other aircraft behind them. Since the original standard does not provide explicit addressing, a send out interrogation will trigger all transponders in range to answer. On the technical side, SSR transponders receive at a frequency of 1030 MHz and transmits on 1090 MHz. While being developed before and in WW II, these two systems (or rather refined versions of it) are still in use

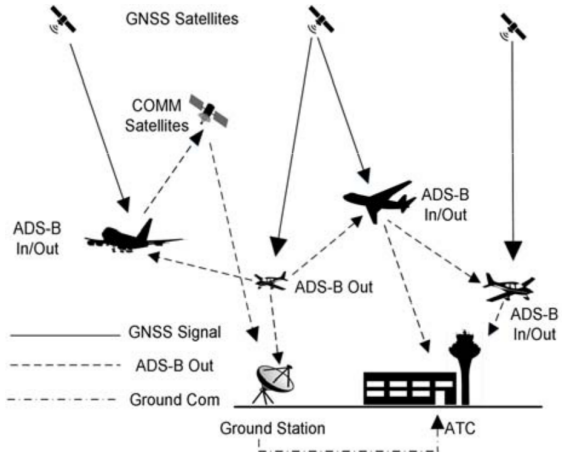

Figure 2.1.: Graphical representation of the ADS-B working principle by Nguyen et al. [38]

and work until now as the only official air traffic control systems.

2.2. ATCRBS Mode S and ADS-B

2.2.1. Principles of Mode S

The ATCRBS Mode S, or mode select, addresses some of the original problems of ATCRBS. Despite being called a mode, it is intended to be a system to replace ATCRBS altogether, while being fully backward compatible with existing ATCRBS technologies [13]. As the name claims, the major feature of Mode S is the ability to interrogate a single aircraft at a time. Every commercial flight is identified by a 24 bits long ICAO (International Civil Aviation Organization) call sign. Modern extensions, like the so called "Extended Squitter", allow MODE-S messages to include other packet types as embedded messages (see Figure 2.2). For this thesis, the packet type of interest are the message formats 17 and 18, the **Automatic Dependent Surveillance - Broadcast (ADS-B)**. For private and commercial operators of high performance aircraft in the United States of America, as well as for all bigger aircraft in Europe and the Asia and Pacific Region, the 1090MHz extended squitter (1090ES) was chosen to be the physical layer link to carry ADS-B data [20, 19, 8].

2.2.2. Automatic Dependent Surveillance - Broadcast

The main feature of ADS-B, compared to traditional SSR, is, as the name claims, the regular, self-reliant (automatic) broadcast of reports. When using 1090ES, equipped aircraft send out 1090ES responses without prior interrogation or pilot inputs. For example, position reports are sent out every 400 to 600 ms. Position reports are based on GNNS signals and therefore, despite the potential inaccuracy of those, a lot more

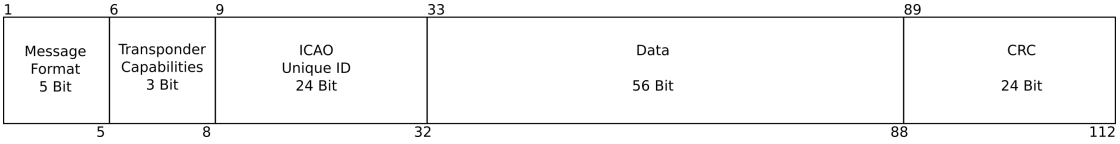

Figure 2.2.: ATCRBS Mode S Extended Squitter downlink format

accurate than positions determined by low resolution PRS. The transmission range of ADS-B messages can rise up to 200 nautical miles (370km).

For commercial high performance flights, the upgrade to ADS-B is easy, since most bigger aircraft already carry SSR Mode S transponders. In 2014, Strohmeier et al. observed that two-thirds of all commercial aircraft send out ADS-B messages [66]. In the Hudson Bay Area in Canada, the full continent of Australia and the North Atlantic ADS-B surveillance corridor, regions hard to keep under surveillance by radar, ADS-B is already operationally used [66].

2.2.3. Security Measures in ADS-B

While ADS-B is a great development for modern air traffic control, it comes with some major security flaws. In fact, there is no security system worth mentioning present [56]. Since no kind of encryption or verification system is used, every ADS-B message and all of its content is freely accessible by everyone in range. Even worse, ADS-B messages can be created and send out with little effort by everyone with sufficient background knowledge and a set of potentially cheap hardware like Software Defined Radios (SDR) and freely accessible libraries [15]. Received ADS-B messages, especially the position reports, can neither be checked for the correctness of the claimed origin (*authentication*), nor for correct and unaltered content (*integrity*). Since there is no possibility to check the position report for correctness, a verification by PSR is still essential. While attacks were assumed to be difficult because of the need of high-end equipment in the beginning, modern research presents a huge set of possible and relatively easy to perform attacks. In [56], Schäfer et al. present a set of passive and active attacks as well. Those active attacks include *Ghost Aircraft Injection*, the injection of a non-existing aircraft into the ADS-B communication channel and *Ghost Aircraft Flooding*, a kind of DoS by flooding the channel with up to 1600 fake aircraft simultaneously. All those possible attacks open up a huge research field to create security measures to ensure the safe and secure use of ADS-B in civilian and military air traffic.

Fortunately, a lot of researchers are working on solutions to the described problems. With passive and active systems as well, security features shall be added to the existing system (see Chapter 3). Unfortunately, since civilian and military aviation is an extremely safety critical working field and a very high number of parties are involved, the introduction of a world-wide system like ADS-B takes very much time and approvals from a lot of involved parties. Therefore, changing the current standard or even replacing it by another, more secure one, seems to be completely unfeasible. Even more, since the security threads are happening right now, it is not justifiable to wait until a new or

altered standard is negotiated.

Therefore, the key requirement for a security system for ADS-B is its backward-compatibility—the original message may not be altered at all. Everyone not involved or not participating in the new security system has to be able to read every signal sent according to the new standard. Additionally, another big requirement is to keep an additional disturbance of the already operational system to an absolute minimum. Since the current ADS-B system is already very crowded and suffers from a lot of message loss [66], an absolute minimum of additional messages shall be used to cause as little disturbance to the operating system as possible.

2.3. The OpenSky Network

As mentioned above, the open nature of ADS-B reports allows everyone with suitable equipment to keep track of every aircraft in range. There are several commercial and non-commercial projects collecting ADS-B data to provide services for a wide spectrum of clients. As an example, the website *flightradar24.com* claims to be the largest ADS-B tracking network worldwide. Started 2006 as a hobby project, Flightradar24 built up a steady growing net of more than 20,000 connected receivers, tracking 180,000 flights per day [47]. Additionally to ADS-B, the service acquires data via multilateration or directly from the Federal Aviation Administration (FAA). Data origin from the FAA are delayed for 5 minutes due to legal and safety reasons. All these flight data are freely accessible by everyone via their website or applications for several operating systems. Big airlines, plane manufacturers and comparable companies can acquire a commercial access to these data. There are commercial competitors to Flightradar24, for example FlightAware. Unfortunately for research, Flightradar24 and its competitors just offer the data as processed information and not the received ADS-B raw data itself. Additionally, accessing large amounts of data in bulk requires commercial access.

To tackle this lack of usable data, the research network *OpenSky* [57] was founded. Based on a growing community of researchers and private volunteers, the association offers free access to a huge base of trillions of ADS-B messages for researchers to identify potential weaknesses and propose improvements to the uprising technology. Initiated in 2012 by researchers of Armasuisse, the Technische Universität Kaiserslautern and the University of Oxford, the network is growing exponentially. In the year of 2016, OpenSky was able to capture about 10,000 messages per second. Until now, this number rose up to more than 300,000 messages per second, provided by over 1,000 ADS-B receivers in 190 distinct countries. In total, the association captured over 11 trillion messages for researchers to analyze so far [53]. Figure 2.3 provides an overview of the amount of already collected messages, the distribution of message types within those messages and the distribution of the feeders to the network around the globe. Projects within the OpenSky community are reaching from performance evaluation of ADS-B, airport optimization up to ADS-B data validation. The data may be accessed as raw data, as it is fed to the network via the receivers, or as so-called state vectors. State-vectors include second-by-second the 3D position, velocity and heading information of every

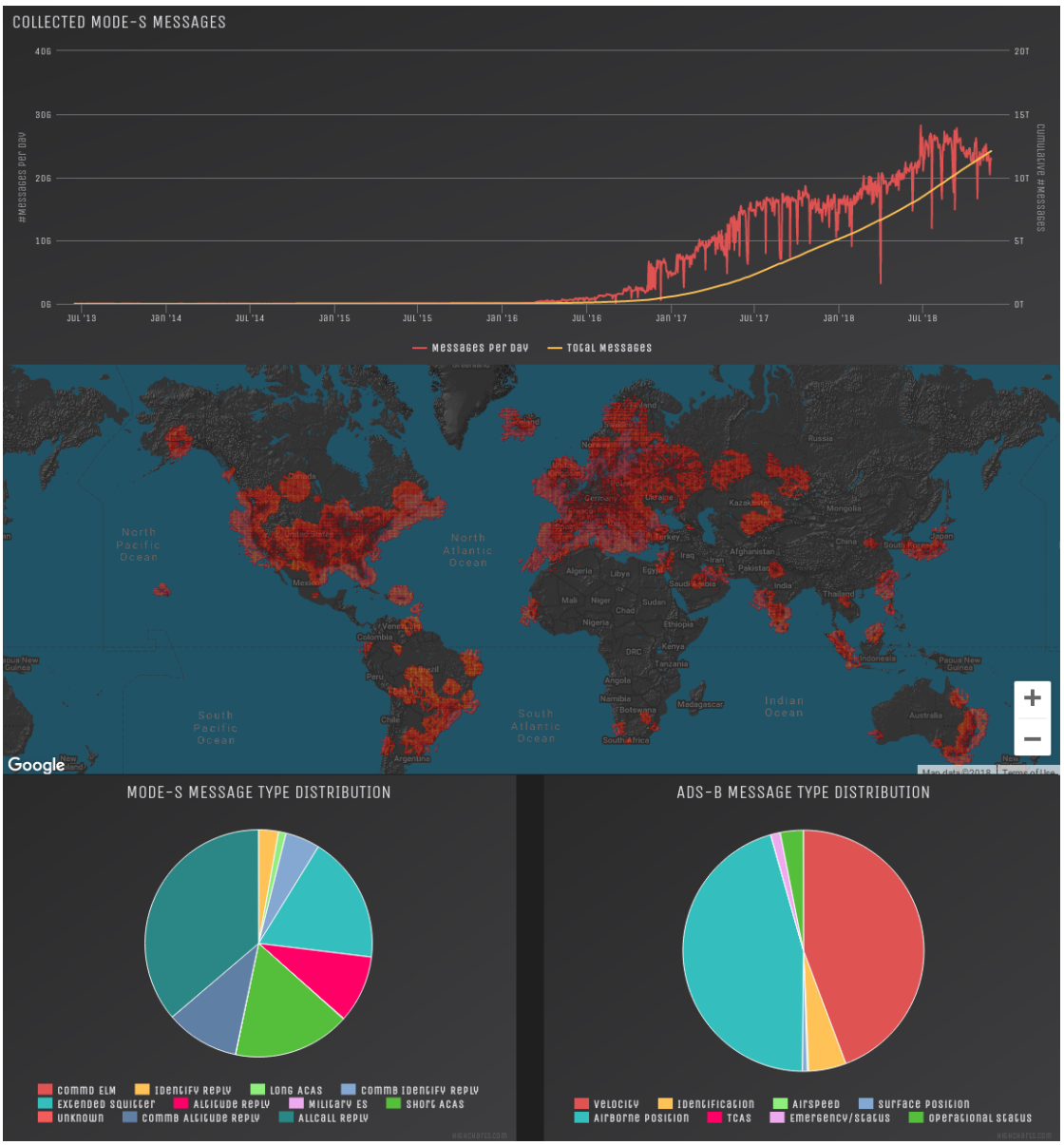

Figure 2.3.: Collected Mode S-messages, network coverage and message distribution types of the OpenSky Network. The colors in the map represent the lowest altitude report in that area by an aircraft. The left pie-chart shows the distribution of downlink formats recorded by OpenSky, the right pie-chart illustrates the message type distribution [53]

tracked aircraft. A collection of state-vectors can be downloaded in CSV, JSON or Apache AVRO data format. The proof-of-concept implementation of this thesis is using up-to-date state-vectors coming from the OpenSky network, allowing a realistic insight on how the concepts will work out when facing "real" ADS-B data that was in fact used in real world.

2.4. Digital Modulation in ADS-B

In simple terms, every data transmission can be broken down to the (wireless or wired) transmission of periodic waveforms, the so called carrier signals. By varying one or more physical properties of the waveform, information can be implemented in the transmission. This process is called *modulation*. In digital signal processing, the most fundamental modulation methods are:

- Amplitude Shift Keying (ASK)
- Frequency Shift Keying (FSK)
- Phase Shift Keying (PSK)
- Quadrature Amplitude Modulation (QAM)

The first three describe methods, where the physical property of the carrier signal is altered to encode information. When encoding a digital bitstream, basically two states are sufficient. For example, with amplitude shift keying, a high amplitude could represent a digital "1", a low amplitude could represent a digital "0". By varying the amplitude in fixed time steps, the bitstream can be modulated. Differing between more than two amplitudes would allow to represent more bits per time unit. QAM is representative for a modulation method, where more than one physical property is used to modulate the signal. Via combining at least two different phases with two amplitudes, more than one bit can modulated at a time. It can be seen as a multiple channel system, with one channel per phase, where each channel uses ASK modulation to transmit information. This information can be either seen as a unit (for example, when in basic 4-QAM a high amplitude at a phase of 0° and a low amplitude at a phase of 180° is present, this could represent the bitstream "01"), or as two independent signal transmissions, that need to be observed separately.

In the case of SSR Mode S 1090ES, Pulse Position Modulation (PPM) was chosen to modulate the signal. In PPM, the carrier signal is partitioned into time intervals. Now, depending on the exact time a pulse is sent within the interval, the received information is interpreted. PPM is usually useful for optical communication systems, since it is, based on the strict timing requirement, quite sensitive to multipath interference. Therefore, it seems unusual to use PPM in a spaciously wireless systems for ATC. But early on, PPM was used for narrowband radio frequency communication because of its simple electrical implementation, allowing small and light-weight receiver and decoder units. Using suitable algorithms, multipath interference can be combined while co-channel

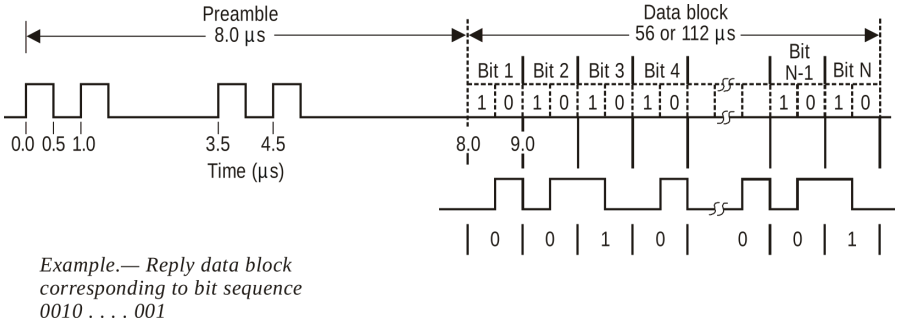

Figure 2.4.: Pulse Position Modulation in Mode S reply according to the standard of the ICAO [49]

interference can be suppressed [71]. Therefore, PPM is known to be relatively robust against interference and collisions [57].

For the 1090ES reply message, the carrier signal is partitioned into intervals of 1 μs. Every pulse has to follow a length of 0.5 μs with a tolerance of plus or minus 0.05 μs. Every message starts with a preamble of 8 μs. This preamble is used to synchronize the receiver on the correct frequency and timing. The pulses have to follow in intervals of 1, 3.5 and 4.5 μs from the first pulse. After the preamble, the data block follows. Here, the 112 bit long ADS-B message can be injected. Every 1 μs interval is separated in two parts. Registering the pulse in the first half of the interval, while there is no amplitude recognized in the second half, the receiver will interpret a logical "1". The other way around, the receiver will recognize a logical "0" (as is shown on Figure 2.4). While transmitting the pulses, the amplitudes shall follow tight restrictions. The variation shall not exceed 2 dB [49]. Technically, the data is transmitted once in its original and once in its inverted form. This allows the signal to resist garbling by other aircraft effectively, since both interval halves would have to be canceled out precisely. Instead, most of the times an interference would confuse both halves partially, causing the system to flag the bit "low confidence" [23].

2.5. Basic principles of cryptography

When it comes to modern cryptography, the most prominent terms are the two basic paradigms: symmetric cryptography and asymmetric cryptography. The two main applications are equal for both paradigms. Cryptography either serves to encrypt the data, meaning transforming readable plaintext to a ciphertext to conceal its content, or to sign data to allow third parties to verify the author and/or the unaltered transmission of the message. Using a secret cryptographic key of variable (bit)length, the message will be fed to a cryptographic algorithm to be transformed into a ciphertext using the cryptographic key. The quality or security of an algorithm is usually measured as *level of security*, expressed in bits. A level of n bits forces an attacker to perform 2^n operations to break the encryption. For unbroken, symmetric cryptographic algorithms without

known attacks, the level of security is usually equivalent to the bit size of the key.

Cryptographic algorithms are usually based on a computational hardness assumption. Candidates are functions, that are relatively easy to calculate for a given set of parameters, but impossible to invert when only knowing the result and the applied function in a reasonable timespan. A classical example would be the *integer factorization*: the composite number $n = p \times q$, where p and q are large primes, is easy to calculate. But only knowing n, for large numbers it is unfeasible to retrieve p and q in a reasonable time, since there is no known solving algorithm for integer factorization that runs in polynomial time [28].

The main difference between the two paradigms are the way the cryptographic keys are handled, as presented in the following two sections.

2.5.1. Symmetric Cryptography

Using symmetric cryptography, the same key is used for encryption and decryption. Therefore, the participants have to find a way to distribute the keys while still keeping them secret. Especially for scenarios, where both parties can not meet in person, this may be difficult. Special key exchange algorithms offer a possibility, but are difficult to perform when there are numerous partners to communicate with. Additionally, key exchange algorithms require mutual communication. Hence, for multicast or broadcast messages, or scenarios where it is not determined who the conversational partner will be, symmetric cryptography is unsuitable for most cases. An additional classical use case for symmetric cryptography is the creation of *message authentication codes (MAC)*. Using the received message and the secret key, a checksum can be calculated and sent along the message. Every receiver knowing the secret key is able to calculate the checksum on the received message by themselves. Only if the calculated checksum matches the one sent along with the message, the message is received correctly and remained unaltered—the integrity of the message is verified. For multicast and broadcast scenarios, symmetric cryptography is hard to manage. Since all involved parties need access to the secret key to decrypt messages or to validate MACs, a sophisticated key infrastructure has to be established, usually leading to a security-through-obscurity like environment. As soon as one of the keys is leaked to unauthorized parties, it can not be used anymore to provide reliable security services. Additionally it has to be mentioned, that symmetric encryption or MACs alone can not provide reliable authentication. Since all involved parties need the symmetric key to validate a received MAC, every party is also able to create valid MACs. A widespread approach to handle these restrictions in multicast- or broadcast-networks is given by the TESLA protocol, that will be presented in Section 2.6.

2.5.2. Asymmetric Cryptography

Asymmetric cryptography relies on a matching keypair instead of a single key: one key for decrypting the message (the *private key*), and one key for encrypting it (*public key*). Asymmetric cryptography algorithms are based on functions, that can not be reverted. To send a message to the communication partner, the message is encrypted with the

Cipher	Block cipher	MAC	RSA	DH F_p	DH (elyptic curve)	ECDSA
Key size	128	128	2000	2000	250	250

Table 2.1.: Suggested key sizes for exemplary cryptographic systems according to the BSI [45] in bit. Key sizes for RSA and Diffie Hellman (based on primes) is only considered secure until 2022, afterwards a key size of >3000 bit is suggested.

partners public key. Since asymmetric cryptography only works in one direction, only the private key corresponding to the used public key can be used to decrypt the message. This comes with the advantage, that the public key can be given out freely or published in key databases. Using asymmetric functions the other way around, messages can be *signed*. Additional to a (public) message, the author calculates a *digital signature* using his private key. In practice, most of the time the signature is a calculated hash-value of the message, that is encrypted with the private key. Upon receiving the message and the signature, the receiver can use the sender's public key to "decrypt" the signature. If the decrypted signature matches the self-calculated hash of the message, the receiver can come to the conclusion, that the message's content was not altered. Additionally to the ensured integrity of the message, the receiver can be certain about the authenticity of the claimed receiver, as long as the identity of the public keys owner is verified by the key distributor and the key was not stolen.

While asymmetric cryptography comes with many advantages, it comes with two major drawbacks compared to symmetric cryptography. First, computational effort when using asymmetric cryptography is in orders of magnitude higher than when using symmetric cryptography. In general, the computational operations for calculating asymmetric encryption are more complex than the operations of symmetric encryption. Therefore, more computational time and computing power is needed for both, sender and receiver of the message [70].

Second, to ensure the same level of security, longer keys have to be used for asymmetric cryptography. As an example, in early 2018 the *Bundesamt für Sicherheit in der Informationstechnik (BSI)* claimed, that there is a need for a 2000 bit long key when using the asymmetric RSA cryptosystem to reach the same level of security compared to a block cipher like the Advances Encryption Standard (AES) with a 128 bit long key [45]. When using modern cryptographic algorithms based on elliptic curve cryptography, the requirement for the key size can be reduced to 250 bit. Despite the fact that longer keys will also require more computational effort when applying the algorithms, especially when dealing with short messages, problems may arise from the longer keys. The amount of data that can be processed and that will be produced by asymmetric cryptography directly correlates with the size of the used key. As an example, using a key size of 4096 bits when using the RSA algorithm, at most 4096 bits of data can be encrypted at a time. The other way around, if the plaintext is shorter than the key size, the ciphertext will end up with a length of 4096 bit anyway. Because of the issues regarding the needed computational power and the ciphertexts length, asymmetric

cryptography is usually only used to encrypt a symmetric key, while the message itself is encrypted with an symmetric algorithm. For signatures, only a hash value of the message is encrypted to overcome the restrictions regarding the length of the plaintext.

2.5.3. CMAC Mode for Authentication

As mentioned above, symmetric cryptography can be used to create message authentication codes. In general, nowadays there are two frequently used approaches to calculate a MAC. The first approach creates *keyed hash based message authentication codes (HMAC)* [29]. Using cryptographic hash functions, like SHA-265 or SHA-3, and a secret, cryptographic key, the MAC is calculated by hashing a special concatenation of the key and the message following distinct rules. The second approach relies on basic symmetric cryptographic functions and a secret key to calculate the MAC. This approach is called *cipher-based message authentication codes (CMAC)* [63].

In general, HMAC algorithms profit from a lower asymptotic cost of their operations and therefore offer faster calculation of MACs for longer messages. On the other hand, CMAC algorithms may be applied using shorter keys to reach the same level of security. The shorter keys will result in smaller blocks needed to calculate the MAC, hence producing less computational overhead. Additionally, many well known block cipher algorithms may profit from hardware support in modern processors or microcontrollers, reducing the calculation overhead once again.

The CMAC Mode for Authentication was published by the National Institute of Standards and Technology in 2005 [18]. CMAC calculations shall be based on well known symmetric cryptographic algorithm like *Triple Data Encryption Algorithm (TDES)* or the *Advanced Encryption Algorithm (AES)*. Depending on the used algorithm, the block size when applying the cryptographic algorithm is decided. For TDES, the block size is 64 bit, AES uses 128 bit long blocks. In general, TDES allows keys with a bit length of 56, 112 and 168 bit. Because of known attacks on TDES, like meet-in-the-middle-attacks or chosen-plaintext/known-plaintext attacks, keys shorter than 168 bit are considered deprecated by the NIST. In general the whole algorithm may be considered broken, since even for long keys the known attacks may bring down the level of security to 80 bits [5]. Therefore, AES shall be used when applying the CMAC algorithm. AES requires keys of at least 128 bit. For key sizes of 192 or 256 bit, AES is still the first and only publicly accessible cipher approved for encrypting information that is classified “top secret” by the NSA [46].

The working principle of the CMAC algorithm is presented in Figure 2.5. Prior to calculation, two secret subkeys are generated. Every round, the matching block of the message will be ciphered with the chosen block cipher algorithm and than XOR’ed to the next block of the plaintext. The result will be ciphered again and then given to the next round. The subkeys will be XOR’ed with the last block of the message to mask the final block. Potential weaknesses introduced by the padding of too short blocks will be intercepted by using a different subkey. The result will be ciphered again and cut the the length T_{len}.

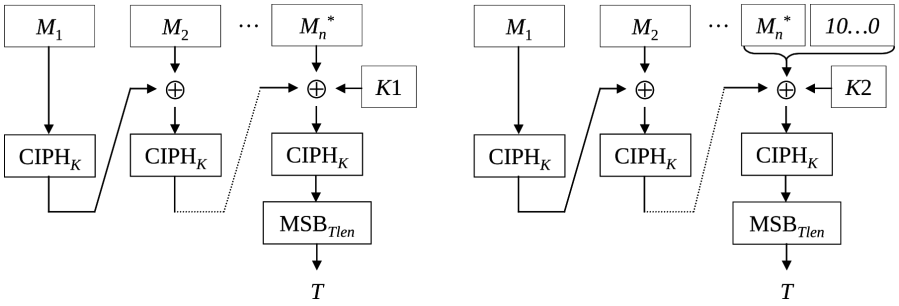

Figure 2.5.: The basic working principle of the CMAC algorithm as presented by NIST [18]. For message lengths matching a positive multiple of the block size of the underlying algorithm, the CMAC calculation is presented on the right side. For messages including parts smaller than the block size, the last block will be padded prior to computation. Depending on the case, another subkey K_i will be used

.

As an input, the CMAC algorithm takes a message of an arbitrary length. Therefore, short messages, smaller than the block size of the chosen cryptographic algorithm, are allowed as well as very long messages. Since the message is split up in blocks of equal length, and these blocks are built "on-line", for longer messages the calculation of the CMAC may already begin before the message is completely fed to the system. As an output, the MAC will be presented as a bit string T with a freely eligible length T_{len}. The length of the output is provided by cutting the calculated MAC to the T_{len} most significant bits. Since the chosen length has no effect on the calculation of the MAC itself, the general quality/security of the output will not be effected in terms of computability, while offering the possibility to adjust the MAC's length corresponding to the needs and properties of the desired system. On the downside, shorter length for the MAC will open up room for guessing attacks. By just guessing a key to calculate the MAC of length T_{len} of an own, fortified message, the probability is 1 in $2^{T_{len}}$ that the MAC will be accepted as valid when using the original key. Additional measures may be considered when using really small bitsizes for T_{len}. The NIST itself recommends a length of at least 64 bit to provide sufficient protection against guessing attacks.

2.5.4. ECDSA

The *Digital Signature Algorithm (DSA)* is an cryptographic algorithm developed by the Federal Information Processing Standard (FIPS) to calculate secure digital signatures. The algorithm uses modular exponentiations and the discrete logarithm problem to create unique, digital signatures using an asymmetric secret key, that can be verified using the corresponding public key. The algorithm itself is similar to the ElGamal signature scheme and was published by the FIPS in [22].

To ensure the security of DSA, very long keys are needed (as it holds for all asymmetric

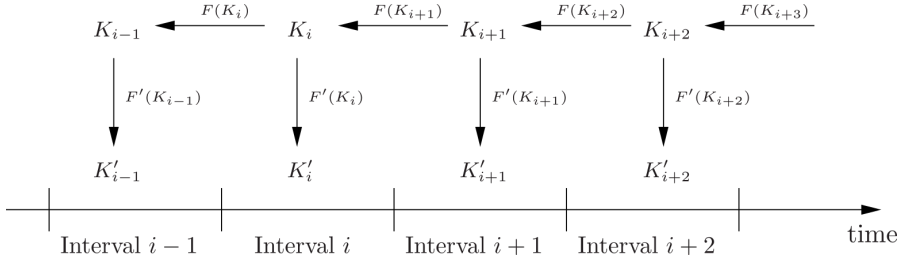

Figure 2.6.: Graphical representation of key chains that are used in TESLA as presented by Perrig et al. [41]

cryptographic algorithms). To tackle this issue, an advanced version of DSA was presented using elliptic curve cryptography in [26]. By the cost of a higher computational complexity, elliptic curve cryptography allows asymmetric cryptography to be used with much shorter keys. To ensure a level of security equivalent to an algorithm using symmetric keys of 128 bit, a key length of at least 250 bits is needed. The signature created will always be four times as long as the level of security. Therefore, digital signatures of at least 512 bits need to be calculated.

2.6. TESLA

The Timed Efficient Stream Loss-tolerant Authentication (TESLA)-Protocol was presented by Perrig et al. in 2002 [41]. In their work, the authors developed a protocol to provide an authentication mechanism for large scale broadcast networks. The protocol itself relies only on symmetric cryptographic MAC functions and a loose time synchronization.

The key to all TESLA-operations are one-way key chains (see Figure 2.6). To create a key chain of length l, a symmetrical key K_l is randomly picked. The key chain will be created by repeatably applying a one-way function F to the key. One-way functions are usually based on cryptographic hashing algorithms like SHA-256, which are not easily invertable. When using long key chains, additional measures may be taken to strengthen the protection of the hashed original value as presented by Caparra et al. in [12]. The used one-way function has to be known by all receivers. The last element created will be the first element of the key chain, denoted as K_0. K_0 will be called the *commitment* to the key chain, since every following element of the key chain can be verified by K_0. For instance, when a node receives the key-chain element at the i-th position, K_i, this element's membership to the key chain can be tested by applying the one way function i-times on K_i. The result has to be

$$F^i(K_i) = K_0. \tag{2.1}$$

Therefore, the publication of the keys will work the exact other way around compared to the generation—that last generated element will be published first, the initial element at last. The authors also discussed on how to store the elements: The whole key chain

could either be pre-calculated and stored at once, or just the initial element K_l can be stored and used to calculate all other elements anew when needed. To find a compromise between computational and storage effort, the authors proposed using an approach by Coppersmith and Jakobsson [14], to reduce the computational and the spacial effort as well to a complexity from $\mathcal{O}(n)$ for the chosen one to $\mathcal{O}(log(n))$ each, where n refers to the length of the key chain.

To use TESLA during the message broadcast, after creating a loose time synchronization between sender and receivers, the time is split in l time intervals of uniform duration. Every element of the key chain is assigned to the corresponding time interval. Since the authors expect a security issue by using the same key for more than one operation, another one-way function F' is created. F' will be applied to every element of the key chain to create an additional key K'_i for every interval. K'_i will be used to calculate MACs for all messages to be sent in time interval i.

The main idea behind TESLA is to provide security by delayed publication of the key. When sending out a message and a corresponding MAC, just the key matching an interval before the current interval will be published. Therefore, the currently received message can not be verified yet, but all messages belonging to this previous interval. The current message has to be buffered until the corresponding key is disclosed. The delay until the key is made public is based on a key disclosure time d, based on to-be-expected clock errors between sender and receiver and propagation delays. Due to the loose time synchronization, a published key can not be used to fool other receivers: By the time the key is revealed, the corresponding interval is over and no further messages using this key will be accepted anymore. Authentication can be checked by verifying that the key is actually a part of the key chain.

This means in detail: Before sending a message M_j in the interval i, the MAC of this message will be calculated by using the interval key K'_i:

$$MAC(K'_i, M_j). \tag{2.2}$$

The sent packet consisting the message M_j will now be created as

$$P_j = \{M_j || MAC(K'_i, M_j) || K_{i-d}\}. \tag{2.3}$$

Upon arriving, the packet will be stored as triplet $(i, M_j, MAC(K'_i, M_j))$ in a buffer, waiting for the corresponding interval to begin when K'_i is revealed.

Up to this point, the integrity of a message can be checked as soon as the key to the corresponding interval is revealed in the following interval. Authentication can be checked by verifying that the key is part of the key chain. But this will only provide the knowledge, that the key and therefore the message is in fact part of the key chain and the author of the message is the one who sent out the first message of the chain. Still, the author's true identity can not be checked, since every participant could have started the key chain.

Therefore, the authors discuss a kind-of Public Key Infrastructure (PKI) application. A certificate authority (CA) shall be build, storing all commitments and therefore first

elements of the key chain for every user. Upon storing, the identity of the user shall be verified. If a user now wants to authenticate the messages of another user, she can contact the CA for the commitment and the key disclosure schedule of this user. By applying the one-way function enough times to the currently received key, the key should finally match the commitment and the user's identity is verified. The contact to the CA itself has again to be verified, therefore every user needs to know the CA's commitment and disclosure schedule at any time.

An interesting point to highlight in this design is the resistance against message loss. Since the key of the previous interval is sent along with every message, even a lost packet will not disturb the process—the key will be sent again in the next package. Even the loss of all packets during one interval, and therefore the loss of all publications of the "piggy-backed" key, can be compensated by making use of the key chain. As soon as the following interval begins, the next key in the key chain will be published. By simply applying the one-way function to this next key once, the lost key can can be recovered. This way, even long periods of message lost (for example caused by strong interference or jamming) can be compensated, as long as at least one later key is received correctly.

2.7. Erasure Codes

In order to protect digital information that is usually stored on a medium that may become faulty over time or that is transmitted over a lossy channel, *erasure codes* may be used. Under the assumption, that the transmitted data may suffer from bit erasures or bit errors, the original message of k symbols is transformed into a longer *code word* with n symbols. The encoding is chosen in a way, that only a subset of the n symbols is needed to reproduce the original message. This subset is usually denoted be k'. The most prominent example for erasure codes could be Reed-Solomon codes [44] (RS-codes), that are used as an error correction method in compact discs (CD), digital audio broadcasting, or several matrix barcodes like the famous QR-Codes. In coding theory, RS-codes as an erasure code are denoted as being optimal. An erasure code is seen as being optimal, if it is possible to reconstruct the original k symbols of the message by successfully receiving exactly k arbitrary symbols of the n symbols sent in total ($k = k'$). On the downside, the operations used in RS-codes for encoding and especially decoding are computationally very complex and do not scale very well with longer messages to be encoded [10]. Modern RS-codes offer encoding with a complexity of $\mathcal{O}(n\ log\ n)$, while decoding comes with a complexity of $\mathcal{O}(n^2)$.

In terms of computational effort, a more efficient class of erasure codes is presented by *tornado codes*. Tornado codes are considered being only "near optimal", since slightly more than k of the n symbols are needed to reconstruct the original symbols ($k' = k+\epsilon$). The deviation ϵ depends on the packet loss patterns and random choices used to construct the code. But by the price of this deviation, tornado codes offer encoding and decoding operations with an complexity of $\mathcal{O}(n\ ln(1/\epsilon))$. According to Byers et al., software-based implementations of tornado codes are about 100 times faster for small symbol lengths and up to 10,000 times faster for large symbol lengths in comparison to RS erasure

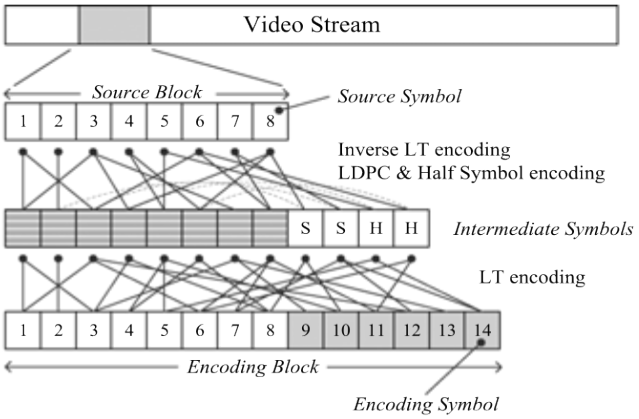

Figure 2.7.: Simplified illustration of creating Raptor Codes presented by Kwon et al. [31]

codes [10]. Using tornado codes, Byers et al. presented *fountain codes*, a class of erasure codes especially designed for multicast and broadcast networks. In fountain codes, a potentially limitless sequence of encoding symbols can be generated from the given set of k source symbols. Any $k+\epsilon$ symbols in this potentially limitless sequence can be used to reconstruct the original message.

One particularly efficient implementation of fountain codes was offered by Shokrollahi as *Raptor Codes* [62]. Raptor Codes are the first known class of tornado codes, that allow encoding and decoding of the message in linear time. The latest generation of Raptor Codes, the *RaptorQ Codes*, offer probabilities of less than 1% of decoding failures when k symbols of the stream have been received. With $k+2$ received symbols the chances of decoding errors drop to less than one in a million [33]. In [31], Kwon et al. present a brief, simplified example of code creation using Raptor Codes to transmit a video stream that is illustrated in Figure 2.7. As shown in the top of the figure, to generate the Raptor encoded stream, the source data is split into source blocks of equal length. Using sequences of XOR operations, the data is converted into intermediate symbols (middle part of the figure). By applying XOR operations one more time, the final encoding symbols are created. In the case of this example, systematical Raptor Codes are used, allowing the original symbols to be part of the encoded block.

3. Releated Work

3.1. Security in ADS-B

Being named the to-be-standard for modern ATC in the USA and Europe in 2010, ADS-B offers a huge potential to increase air traffic *safety* for all participant. However, researchers began early on to identify major *security* issues. One of the pioneering works in this field was presented by Costin and Francilion in 2012 [15]. The authors identify

- the lack of authentication mechanisms to protect against message injection,
- the lack of signatures or MAC to protect against tampering,
- the lack of message encryption to protect against eavesdropping,
- the lack of challenge-response mechanisms to protect against replay-attacks and
- the lack of ephemeral identifiers to protect against privacy tracking attacks

as the main problems based on missing security features in ADS-B. Summarizing, they claim that "given the budget [of more than $1176M in the USA alone] involved, and the sensitivity of air-traffic, it is surprising that such a system was not designed with security in mind" [15].

Based on this and comparable work, subsequently Schäfer et al. presented a set of experiments in real-world environments in [56] to prove the feasibility of the considered attacks. They conclude, "that without appropriate countermeasures, critical air traffic management decision processes should not rely on ADS-B derived data".

Fortunately, since this time a number of researchers presented several proposals to tackle these problems. These proposals may be grouped in two classes. The first class of proposals describe *passive* solutions, mostly using the physical properties of the transmissions to verify the claimed origin of the signal (either in terms of either authentication or forgery). The other class includes *active* solutions, where the ADS-B signal itself is altered or extended to transmit additional information to fulfill security goals.

3.1.1. Passive Approaches to Secure ADS-B

Several approaches in terms of passive solutions were made to mitigate the threat of the injection of fraud position reports via multilateration (MLAT) [55, 39, 34]. These approaches usually base on a number of receivers (for traditional MLAT algorithms at least four), that measure the time difference of arrival (TDOA) of a specific signal. Using these time differences the claimed position of the sender may can be verified by

intersecting the possible transmission origins. In [60], Schäfer et al. criticize the need for a tight time synchronization for traditional MLAT operations and present an alternative mechanism without the need for synchronized receivers.

However, while still acknowledging the effectivity of MLAT solutions, Strohmeier et al. identify the problem, that these solutions will only have an effect in areas providing a reliable and thorough coverage [64]. Therefore, the authors see MLAT solutions only as an option for crowded airspaces around airports. As an alternative, they present two continuative solutions. The first one uses statistics and TDOA based on only two receivers to detect falsely injected data with a high certainty. The second approach uses at least three receivers to not only detect the injection of fraud data, but to also estimate the correct positions the data was injected from. Additionally, the authors claim that their solutions provide faster and more reliable results compared to MLAT solutions.

Summarizing, via MLAT solutions a received position report can be verified, as long as a sufficient coverage of additional receivers is given. Since position reports are the most crucial information provided by ADS-B, a kind of integrity-protection is secured. But it has to be noted, that even advanced systems rely on a sufficient coverage and a trustworthy platform and can not offer authentication. While it can be verified if the aircraft really is at the position is claims to be, it can not be checked if it really is the aircraft it claims to be.

An approach to provide authentication with a passive system by fingerprinting link layer characteristics of the aircraft transponders was presented by Strohmeier and Martinovic [65]. The authors found that it is possible to exploit implementation differences to identify different types of transponders used in aircraft. But it was also observed, that aircraft types or aircraft fleets all use the same transponders, making it impossible to allow a fingerprinting of one specific aircraft by this technique alone.

3.1.2. Active Protection Against Adversaries

Several researchers tackled the security issues of ADS-B by altering or extending the data to be transmitted. Mostly, this is done by using symmetric or asymmetric cryptography. Despite being discussed by some researchers [21, 2, 15], an encryption of the ADS-B signal to ensure integrity, authenticity and to protect against eavesdropping would be a possibility, but is undesired and quite unrealistic for several reasons (as discussed in Chapter 1 and Section 2.5). Instead, a lot of research focuses on using message authentication codes or digital signatures to provide integrity or authentication for ADS-B messages.

Cryptographic Solutions for ADS-B via UAT

In [40], the authors discuss the usage of asymmetric elliptic curve cryptography to provide security for ADS-B messages. The authors of [3] presented a framework based on online/offline identity-based signatures. But unfortunately, these publications are based on ADS-B transmission via Universal Access Transceiver (UAT), as it is used as the physical layer link for ADS-B messages in general aviation in the USA. General avia-

tion is usually used as as a generic term for all private aircraft that will not exceed a altitude of more than 18,000 ft. UAT provides longer messages compared to 1090ES and was especially designed for transmitting ADS-B and additional services. Additional to ADS-B information, UAT is able to carry more information to provide safety and informational services to the pilots. The Flight Information Services-Broadcast (FIS-B) provides weather texts, weather graphics or Automatic Terminal Information Services (ATIS). It is restricted to areas that have ground surveillance infrastructure. A second service attached to UAT is the Traffic Information Service-Broadcast (TIS-B). TIS-B provides a TCAS service using the same system ADS-B is using to allow existing TCAS devices in the aircraft to be replaced. But since UAT is fixed to a very special subset of aircraft (geographical and field of application as well), these solutions will not match to the given problem dicussed in this thesis.

Because of the limited room for data transmissions using 1090ES and respecting the limited computational power of the transponders, most research in this field focuses on using message authentication codes based on symmetric cryptography or hashes.

Authentication and Integrity via HMAC

In [27], Kacem et al. presented a solution based on keyed-hash message authentication codes (HMAC). Securing only the 56-bit long payload of the ADS-B message itself, the authors propose to gather at least six ADS-B messages. The 128-bit long HMAC is applied on the assembled payloads of all messages. Next, the HMAC is split again in six parts, each smaller than 24 bit. To reconstruct the correct order, an ID is attached to the HMAC parts. Now the singe HMAC parts are assembled to the ADS-B messages by replacing the CRC-field. To solve the issue of the key distribution, single-use or short-time keys are used: as soon as an aircraft approaches an airfield and is granted permission, it will be presented a set of keys. For each zone under the control of an ATC tower in the aircraft's path, one key will be given out. When transiting a zone, the aircraft chooses the corresponding key.

Based on the hitherto addressed limitations, some major issues have to be discussed in the solution by Kacem et al. First of all, since an HMAC is suitable for providing data integrity as well, the replacement of the CRC-field of the ADS-B message could seem to be a possibility at first. But in contrast to CRC, the HMAC offers only a possibility to detect bit errors, while CRC comes with the possibility to repair them in some degree. Additionally, the ADS-B standard itself would have to be changed, since legacy-devices would no longer accept these altered messages because of the non-matching CRC. To calculate the HMAC, six messages will have to be restrained and send out later, making every position report (which are sent out twice per minute) at least three seconds older. Assuming a flight velocity of up to 900 km/h for a modern airliner, this could result in an error of up to 750 m, tackling the real-time capabilities of the ATC. As long as the ATC-tower infrastructure is trustworthy, the usage of short-time keys could solve the problem of the key infrastructure. Unfortunately, the possibility to verify of the messages would be restricted to the ATC-Tower corresponding to the chosen key. Additionally, the approach presents no solution how to distribute the keys securely, opening up the

start-up problem anew, since authentication can only be provided, as long as the party handing out the keys can be sure of the aircraft's identity. Last but not least, the system could suffer a lot from package loss, since all transmissions including the single parts of the HMAC have to be received correctly to allow a assembling and validation. As it will be presented in Chapter 3.2, the necessary reliability may not be given in real world.

Multicast Message Authentication Codes

A fitting, alternative solution to get the key management under control could be offered by Multicast Message Authentication Codes (MMAC). A pioneering work regarding MMAC was presented by Desmedt et al. in 1992 [17]. A general, basic idea for MMAC would be to share a distinct key to every possible receiver beforehand. When sending the message, a MMAC is constructed by concatenating HMACs of the message, one for each receiver. Every receiver can pick the MMAC part matching their key to validate the message. This comes with two major problems: First of all, a secret key exchange has to be done with every receiver. Second, the MMAC scales linearly with the number of receivers. An optimized alternative was presented as κ-secure MMAC. A MMAC will be called κ-secure, if at least κ receivers are needed to construct an own MAC that can fool another receiver. In 2001, Carnetti et al. presented a scheme to construct κ-secure MMAC by concatenating many pseudorandom functions whose output is a single bit [11]. Research by Boneh et al. presents the need for an advanced digital signature design to build short and efficient MMAC in [9]. But still, while reducing the length needed for a secure MMAC drastically compared to the initial approach, the length of a MMAC for an unknown number of possible receivers might still not fit the low possibilities offered by 1090ES. Additionally, the key distribution issue still persist.

SAT

In Section 2.6, the TESLA protocol was presented. Using key chains and a loose time synchronization, authentication mechanisms for broadcast network are built up completely based on symmetrical cryptography. But as Section 5.2 will show, the basic protocol will not fit to the ADS-B environment by default. A series of compromises and commitments will have to be made to adapt the protocol to the given problem.

One approach to make these adaptions was presented by Berthier et al [7]. In their work, the researchers present a new, adapted protocol, called Security in the Air using Tesla (SAT). By relying on a PKI, a certificate given out by a CA is used to reliably validate the first (or in case of a running key chain the last sent out) key of the key chain. Since every following element of the key chain can be validated via the first element, the complete following key chain can be validated. To transmit the additional information, the authors decided to just append them on the original message. Since every legacy receiver would just stop interpreting the message after the CRC of the 1090ES packet, only receivers aware of the security system would mind the appended parameters. This way, the protocol itself will not be changed. As presented in Chapter 2.5, secure symmetric cryptography still requires key sizes of 128 bit and higher. Since

SAT relies on hash based MACs, even longer keys are needed to achieve the same level of security. Since TESLA requires the key to be send along with the message, the size of the 1090ES transmission would be increased by more than 200% by attaching the key alone. Additionally, the MAC and a sequence number have to be attached and the certificate along with the last used key has to be sent out periodically. Making several compromises regarding the duration of one interval, the frequency of publishing the keys and the certificate broadcast period, the authors claim to be able to reach an supplemented overhead of 45 bits per broadcast, extending the original message by 40%. In crowded areas, like airports, this overhead could significantly increase the already high package loss. Not considered was the to be expected package loss, that could render transmissions of the interval key or the certificate useless, since no error correction method was applied. Additionally, the protocol might not withstand replay attacks.

3.2. Second-Channel Communication for Additional Data Transmission

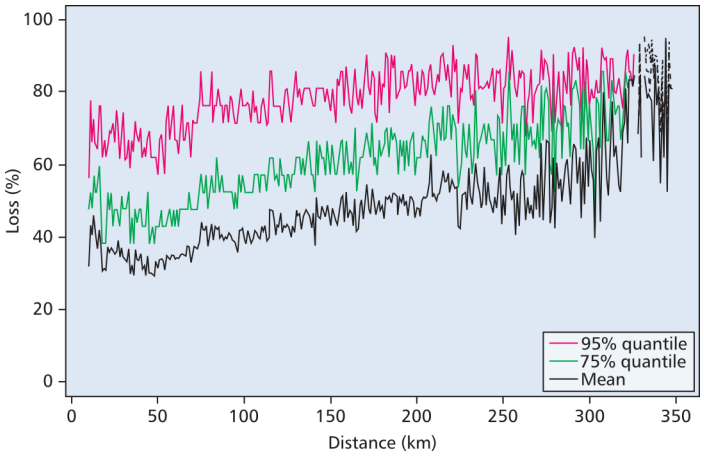

Figure 3.1.: Real world package loss of ADS-B messages as observed by Strohmeier et al. in [66]. Even when observing aircraft nearby (up to 50 km), the rate of ADS-B messages successfully received is up to 40% lower than it should be corresponding to the message publication schedules. For aircraft farther away, a linear increase in this message loss has to be expected.

As already mentioned, changes to the ADS-B protocol itself are out of discussion when implementing a security model. Appending additional data to the message itself, as it was presented in [7], may also be problematic. In [66], Strohmeier et al. evaluated a lot of data gathered by OpenSky to make some observations regarding the performance of ADS-B in reality. Especially the observations regarding message loss rates and channel

quality are worth mentioning. Even for aircraft within a range of 50 km, the authors observed a package loss of up to 40% (see Figure 3.2). For higher ranges, a linear increase could be observed, especially in crowded areas. It has to be assumed, that especially message collisions are the main reason for the loss. Extending the messages would provide even more potential for collisions. Therefore, a main principle in this thesis is to add as little further traffic to the system as possible.

The physical layer link using pulse position modulation of 1090ES is surprisingly simple. Since the only kind of modulation used is based on the amplitude and timing, it is worth considering additional data transmission by combining the pulse position modulation with an additional kind of modulation.

In [43], the authors research the possibility to hide additional information within the preamble of an 802.11 WiFi signal by using shifts in the time domain and phase rotations in the frequency domain. For the bandwidth of 20 MHz the authors were able to embed up to 8 additional bits. As will be presented in Chapter 4, this idea could be adapted to an 1090ES transmission to win a few additional bit.

Mitchell et al. presented the possibility for high data rate transmission in Ultra-Wideband communication by combining pulse position modulation with pulse shape modulation [37]. By using mutually orthogonal pulse shapes they found a possibility to add additional information to a classical pulse modulation. Since pulse shape modulation is designed for wideband communication, further research would be needed to analyze the capabilities of the ADS-B signal to include additional information via the shape of the pulses.

Comparable research was done by Liu et al. [32]. The authors combined pulse position modulation with frequency shift keying to increase the data rate in optical transmissions. By additionally using phase modulation, the data rate was increased even further.

In fact, an additional kind of modulation is already used in 1090ES. The data pulse of the 1090ES interrogation uses 180-degree phase reversals to provide additional resistance against corruption [48, 49]. Therefore an additional data layer provided by phase modulation may be a promising approach in the 1090ES responses too.

In [42], the authors proposed to combine the pulse position modulation with an additional 8-PSK to win three additional bits per transmitted bit for additional information. Two years later, Yeste-Ojeda and Landry went even further and proposed D16PSK for one additional layer of information [72], despite the sensitivity of higher level phase modulation for interference by other transmissions. This idea was researched further by Nguyen et al. in [38]. The researchers claimed to be able to provide four times the additional bit size for additional information. According to the authors, these bits shall be used for adding a digital signature based on the Advances Encryption Standard (AES) along with the message. Since AES is symmetrical block cipher, the authentication problem will not be solved at this point. Via Reed-Solomon codes, the correct reception shall be secured. While Nguyen et al. were able to present the feasibility of their approach using hardware-in-the-loop simulations, unfortunately none of the researchers could provide real world tests. Additionally, securing every single message with a digital

signature based on asymmetric cryptography would set the transmitters under heavy load. Therefore, the real-time capabilities of the system could be endangered. Further research is needed to prove the feasibility of using higher level phase shift keying without losing to many messages due to interference. Additionally, the in-use transponders need to be analyzed about their capability of dealing ongoing heavy loads caused by the mathematically complex cryptography operations.

4. Concepts and Approaches for Second-Channel Communication

As presented in Chapter 3, the transmission of additional bits by hiding information within the existing ADS-B signal may be the best approach to face the challenges of the already crowded baseband. At the time this thesis is presented, the exploration of the feasibility of different approaches of information hiding in ADS-B is part of another project in the Distributed Computer Systems Lab at the Technische Universität Kaiserslautern and therefore out of the scope of this thesis.

Nonetheless, the decision which technique to use for information hiding and the possible amount of bits offered along with this decision will mainly decide the design of the security model. As presented, secure communication that can withstand the technical possibilities in modern computer science requires thought-out concepts and in the end an adequate number of bits to transmit the information needed to secure the signal. Since the size of only 112 bit for the 1090ES message itself is really short, the possibilities to include a higher number of additional information are very limited.

For this reason, a number of possibilities to transmit these additional information are presented and discussed in this chapter. Even though there is no concluding result on how much information may be hidden in a real world transmission of 1090ES messages until now, the best candidates may be identified for further research. Based on the expected possibilities offered by these solutions, designs on how to realize a security model with the given possibilities will be discussed in Chapter 5.

4.1. Mode S - Physical Layer Overview

In [49], a set of restrictions for a standard-compliant transmission of Mode S messages is given.

The uplink transmission, comprised of interrogations, are settled on a baseband of 1030 MHz. Since the information transmission via ADS-B messages is done automatically on a regular basis, Mode S interrogations are no longer needed for this protocol. Nonetheless, the Mode S interrogation guidelines offer an insight about the physical possibilities offered by the baseband, since they can be rated as proven and tested based on their more than 30 year long operation.

The carrier frequency will be, similarly to Mode S responses, pulse modulated. As shown in Figure 4.1, the preamble consists of only a pulses P_1 with a duration of 0.8 μs, followed by a second pulse P_2 with the same duration 2 μs after the start. Mode A and C transponders will interpret these pulses as coming from an antenna sidelobe and therefore not respond. The following data block of 56 or 112 bit consists of only one

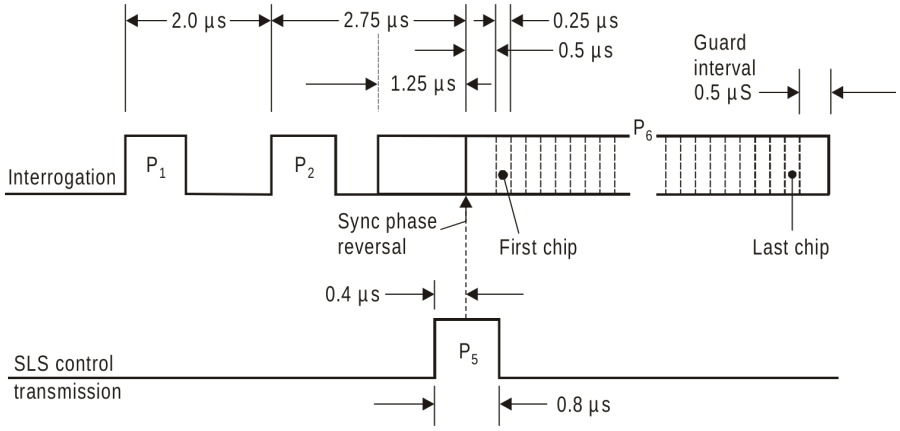

Figure 4.1.: Mode S interrogation pulse sequence as standardized by the ICAO [49]

pulse of 16.125 μs or 32.25 μs duration. After a phase reversal synchronization the data itself will be modulated via an internal phase modulation. Subsequent phase reversals indicate a data bit of 1, no phase reversals indicates a data bit of 0. With phase reversal positions with an interval of 0.25 μs, the Mode S interrogation offers a data rate of 4 MBit/s and therefore transmits data on a four times higher rate than Mode S replies.

The Mode S reply, the actual signal of interest, pulse sequence was presented in Figure 2.1. The reply is sent out on a baseband of 1090 MHz with a generous tolerance of plus or minus 1 MHz (as an comparison, the Mode S interrogation is bound to a strict tolerance of 0.01 MHz according to the standard [49]). The signal is pulse position modulated, with pulses of 0.5 μs in 1 μs intervals. The preamble consists of a sequence of 4 pulses. All reply pulses shall start at a defined multiple of 0.5 μs from the first transmitted pulse. A tolerance of plus or minus 0.05 μs is considered. The pulse amplitude variation shall not exceed 2 dB. The required spectrum limits for transponder transmitters are presented in Figure 4.2.

4.2. Candidates of Information Hiding

4.2.1. Preamble Gaps

Inspired by [43], the possibility to hide additional information within the gaps of the preamble was discussed. The basic idea is to insert additional pulses with an amplitude lower than the minimum triggering level (MTL) of standard transponders. Since only four of the eight preamble intervals are filled with preamble pulses, the four other intervals could be used to transmit additional pulses. Since the pulses have to be sent with an amplitude below the MTL, the signals could actually be sent out with a higher frequency without interfering with other transmissions, since they would still follow the spectrum limits presented in Figure 4.2. This solution comes with some major drawbacks. The

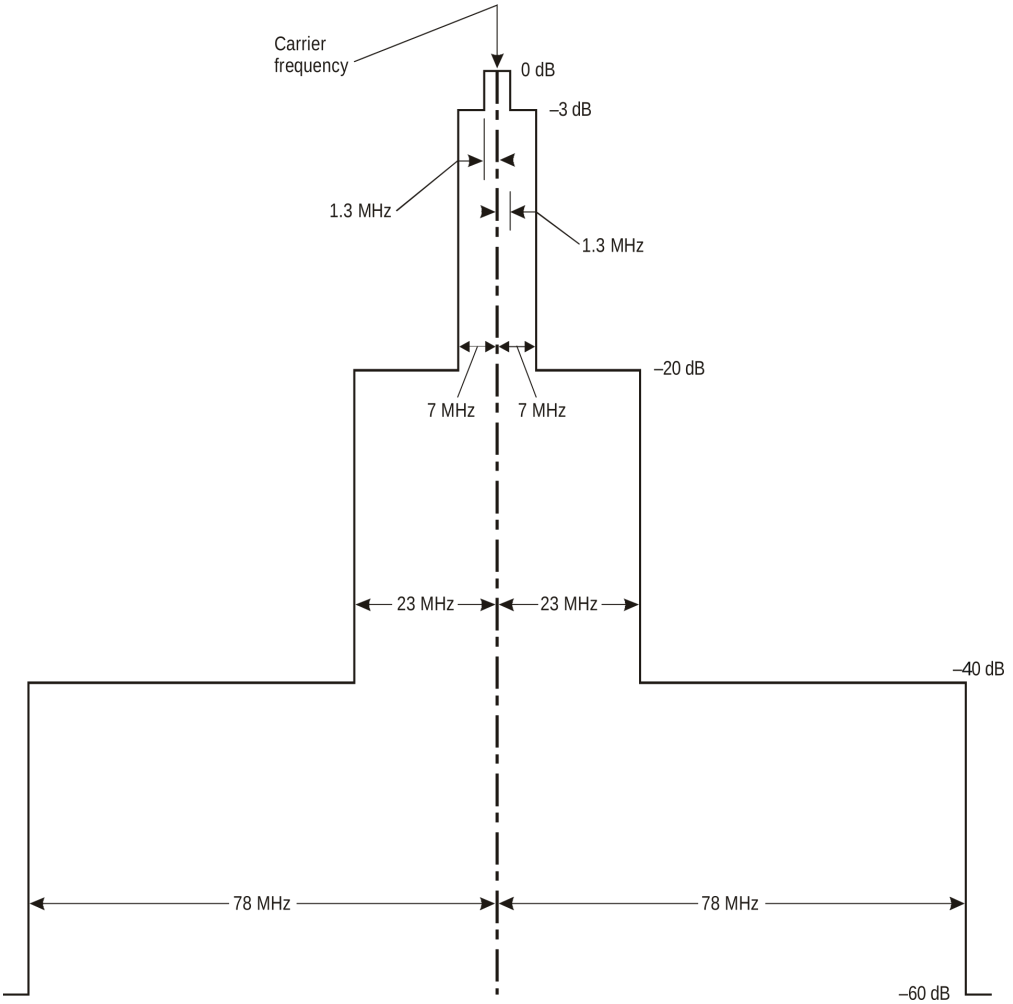

Figure 4.2.: Required spectrum limits for transponder transmitter as standardized by the ICAO [49]. The standard allows a tolerance of 1 MHz, therefore the spectrum may shift in its entirety plus or minus 1 MHz along with the carrier frequency

preamble itself is used to synchronize sender and receivers in terms of timing and frequency. Any disturbance could make the synchronization fail. Therefore, it would have to be made sure, that the signal is sent with amplitudes below the MTL of any possible receiver. This could be hard to realize in reality. Additionally, since now the signal would not trigger the original transponders any longer, additional hardware would be needed to catch the signals. Because of the lower amplitude, the signal's transmission range would be shorted, making it impossible to use the additional authentication information of aircraft far away. On the upside, if there would be a possibility to use additional hardware to interpret the signals in the preamble gaps, the method could be spread to all gaps in the Mode S transmission, since there will be a gap in every interval. But

because of the high requirements, the feasibility of this idea is rated very unlikely.

4.2.2. Amplitude Shift Keying

A simple method to introduce an additional modulation to the existing one could be amplitude shift keying. Every pulse could be sent out with a higher or lower amplitude, with a lower amplitude representing a data bit "0" and a higher amplitude representing a data bit "1" in the additional message. This way, the data rate could be doubled, since every pulse would represent two bits. But as mentioned above, the possibility of amplitude shift keying would be very limited, since the variation of a valid signal shall not exceed 2 dB. Further research would be needed to ensure that a amplitude variation this low could really be recognizes reliably by the used receivers to add reliably recognizable data. Additionally, the added information would be very sensitive to noise.

4.2.3. Frequency Hopping

Comparable to the amplitude shift keying, the frequency of the pulses could be varied pulse by pulse to include additional information. Again, a pulse sent out with a lower frequency could be interpreted as a data bit "0", a pulse sent out with a higher frequency as a data bit "1". Based on the sensitivity of the transponders, even more different frequencies could be used to hop in, allowing more than one bit to be modulated at a time. Additional to the allowed 1.3 MHz frequency deviation presented in Figure 4.2, the standard allows 1 MHz tolerance in any direction, opening up the space for even more possible frequencies to hop. Further research would be needed to analyze the sensitivity of the aircraft transponders to frequency shifts and the impact of blurring the frequency spectrum when starting to shift on the reliable demodulation. Due to the varying quality of the transponders equipped in aircraft, the center frequency is often shifted in real world. An implementation based on frequency hopping would have to make sure, that the frequencies to hop to will not leave the given spectrum limits.

4.2.4. Phase Shift Keying

A very promising approach is the injection of bits via phase shifting within the pulses. Since phase shifting is being used successfully by Mode S interrogations, the existing hardware used in SSR Mode S environments should be able to handle phase shifts by 180-degree out-of-the-box. A lot of hardware used in the ADS-B system is using already existing SSR Mode S receivers directly. But for aircraft later equipped with hardware especially designed to work with ADS-B, the possibility to receiver and react to traditional SSR Mode S interrogation may not be given. For those transponders, additional research is needed to learn about the suitability to handle phase shifts.

The main difference of Mode S responses to interrogations may be, that no continuous pulse is used to transmit the signal, but a lot of small pulses instead. Therefore, there have to be new restrictions when to shift the phase to add information. Mode S interrogation uses intervals of 0.25 μs for phase shifting. A standard pulse of Mode S replies is exactly twice as long. Therefore, a possible phase shift right in the middle of a

pulse could be considered. A phase reversal present in the middle of the pulse could be interpreted as a data bit "1", no phase reversal could be interpreted as a data bit "0". To allow the usage of phase reversals as information carrier, a synchronization similar to the Mode S interrogation is necessary. Since no reserved space for synchronization is provided, another solution has to be found. As this point one could exploit the fact, that every Mode S reply message has to start with a five bit long *Message Format*-, or *Downlink Format*-field (cf. Figure 2.2). For ADS-B, the message format is 17 or 18, therefore a binary 10001 or 10010. These five bits will be sent at the beginning of every ADS-B message. Since a data bit "0" pulse will fill the second half of the interval and a data bit "1" the first, the last/second last two bits will build a combined 1 μs long pulse. This pulse could be used to synchronize sender and receiver, at the cost of 5 potential additional bits. Every pulse following the first five pulses could be used to transmit additional information. It has to be considered, that the pulse position modulation was chosen because of its robustness against interference that do not precisely hit pulses or, to be precise, the absence of the pulses (as mentioned in Chapter 2.4). Since an additional phase modulation would not profit from described "dual transmission" of the message, the consequence of interference to the phase modulation has to be examined further.

5. Designing the Security Model

As explained in Chapter 4, the design of a secure, active verification system highly depends on the possibilities offered by the additional data transmission. As a rule of thumb, one could say, that the more bits are offered, the better (up to a certain level). But since there is no concluding result on how many bits can be hidden reliably in the ADS-B messages until now, the goal of the following chapter is to find a secure and stable solution offers the possibility to scale with the results of further research. The solution presented in this chapter is based on the promising second-channel transmission via basic phase shift keying. Therefore, for every bit of the original ADS-B message, one additional bit is assumed (where most can be used for securing the communication, while the first five bits are required for synchronization). In case further research will restrict the amount of bits to a way smaller or higher number, the proposal is built to scale with these limitations. The corresponding alternatives as well as the proposed parameters used in this solution will be discussed in Chapter 7.

5.1. Security Model Requirements

Summarizing the problems and information given in the previous chapters, a security model to protect ADS-B should offer the following properties:

- **Legacy support**: The protocol defining the packet format and content of ADS-B messages shall not be altered. Since the standardization of ADS-B took a long time and a lot of parties are involved, it is highly unrealistic to face the described security problems on short-term by designing a new or altered standard. Parties not using or not involved in the proposed security model shall still be able to use ADS-B as standardized now.

- **Low disturbance**: The band ADS-B is operating on is already very crowded and suffers from a lot of interference and packet loss. Especially in dense areas with a lot of expected aircraft sending out their periodic messages at the same time (for example airports), the successful reception of the single messages can not be guaranteed. But especially these areas are the once that need a stable and reliable reception of the messages at most, since the reports are crucial for a safe air traffic control. The security system build on top of ADS-B shall disturb the existing transmission on the lowest possible level. Additional crowding of the band, by increasing the number or amount of transmissions on the same frequency or additional interference caused by the second-channel transmission, shall be prohibited as far as possible.

- **Integrity of the message**: The ADS-B message content has to be protected against forgery. The receiver of the message needs a possibility to verify that the messages content was not altered during transmission or at the receivers end.
- **Authentication of the author**: By verifying the identity of an ADS-B message, unauthorized injection of messages or identity theft may be prohibited. The security model shall ensure that no third party is able to send realistic-looking messages in the name of another aircraft or to inject messages from non-existing aircraft/call signs.
- **Computational effort**: Since equipment used for ADS-B transmission and reception, especially the transponders in the aircraft, are usually limited in terms of computational power, and ADS-B reports are sent out very frequently, the security model shall not set the receivers under high, avoidable load.

Reviewing the information already gathered, a solution based on symmetric cryptography could be the best fitting option. To reach a reliably high level of security, only comparatively short keys are needed. This property will be made used of when designing the security model, since the key has to be published eventually to allow receivers a verification of the messages. Correspondingly, the encrypted messages itself or the associated message authentication code can be short while still being hard to counterfeit. Additionally, a strong argument for the usage of symmetric cryptography is the low mathematical complexity of the used algorithms compared to asymmetric algorithms.

Related work is often based on HMAC, as presented in [29]. This thesis proposes the usage of CMAC based on the AES as presented in [63] instead. A general introduction to the CMAC algorithm is given in Section 2.5.3. HMAC algorithms, being developed first, are usually preferred since hash functions are faster to calculate than block ciphers in general. But this impression is usually based on the better asymptotic costs of the HMAC algorithm over the CMAC algorithm. For short messages, the overhead that comes with HMAC will most likely weight heavier than the possible savings [16]. Since the size of a 1090ES/ADS-B message is even smaller than the standard block size of AES, one could hardly profit from potential savings when using HMAC. To calculate a secure MAC, CMAC based on AES may be used with smaller keys compared to HMAC based on SHA-256 or comparable algorithms offering the same level of security. Additionally, modern processors and microcontrollers mostly come with hardware optimizations for standard block cipher encryption algorithms like AES (for example by offering dedicated AES opcodes), making the usage of AES even faster and more power-saving [1].

But after all, as already discussed in Section 2.5, symmetric cryptography alone can only be used to ensure the integrity of the messages, and not to provide reliable authentication. At this point one has to still rely on asymmetric cryptography. Using digital signatures and open but securely administrated key-servers or a trusted PKI, authentication can be provided. To get a hold on the high amount of data needed to provide reliable signatures, a TESLA-inspired system shall be established. In the proposed design, asymmetric cryptography is used to provide a digital signature when publishing the key used for creating the MAC. Since the keys and the corresponding signatures are

not based on real-time flight data, they might be pre-computed. The correct reception may be ensured by using Raptor Codes/fountain codes [62].

5.2. TESLA for ADS-B

The TESLA protocol offers great possibilities for providing authentication and integrity in broadcast environments. But in its basic design, the use case does not exactly fit the given problem when considering ADS-B. Several commitments have to be made to match the protocol to the given problem:

1. TESLA, and especially the PKI application of it, requires a loose time synchronization between sender and receiver. This synchronization shall be achieved via a simple two-round time synchronization protocol. ADS-B does not offer time synchronization, since no timestamps are used within the broadcast, and neither does a mutual communication exist.

2. Since every message sent when using TESLA needs to include the MAC of the current message and the key of the previous interval, there is a need to transport a lot of bits additionally to every message. To achieve a sufficient level of security, the key alone should consist of at least 128 bits when using cipher based MAC (referring to the recommendations of the BSI mentioned in Section 2.5). When using hash based MAC, modern algorithms like SHA-256 require an additional 128 bits, adding up to a total of 256 bits. When implying the possibility to send an additional bit for every bit of the original ADS-B message, not even the key for a CMAC-based MAC could be sent along with the message.

3. TESLA may put the receiver under high computational load when joining the reception at an advanced time of the key chain. In the worst case, the receiver starts at the last element of the key chain, forcing it to apply the one-way function the full l times to reach the commitment-key before the authentication can be validated. Since aircraft sending out ADS-B messages move at a high speed, it is very likely that the first element of a key chain observed is not the actual first element of the key chain. Therefore, the intervals for initiating new key chains would need to be small. In turn, the commitment message stored at the CA would need to be updated a lot.

4. The realization of the described PKI could be very difficult for a world wide net as it is used for ADS-B. Since the commitment keys and the key disclosure schedules have to be updated with every change in the key chain, all aircraft would need a permanent connection to the CA to update their own information and to receive the current updates of other aircraft. This requires a lot of communication and it may be difficult to realize a world wide operating, permanently reachable and up-to-date CA, that can deal with the fast moving changes required during operation.

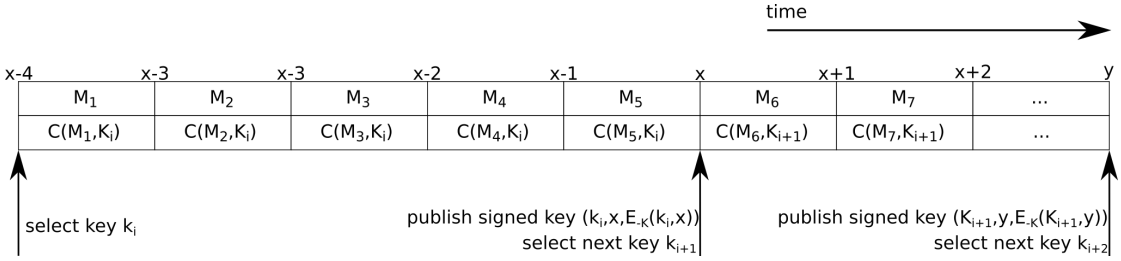

Figure 5.1.: Basic overview of the proposed security model. A CMAC will be attached to every message sent using a secret key K_i to provide an integrity check of the message. At the end of the current interval, the key will be disclosed. The transmission will be signed by the aircraft using asymmetric cryptography and its private key, denoted as $-K$, to provide the possibility to verify the aircrafts authentication

All of those drawbacks prohibit a direct integration of the protocol to ADS-B. But accepting a few compromises, the basic mechanics of TESLA can be adapted to create a new protocol, closing the mentioned gaps.

5.3. Adapted TESLA

5.3.1. Overview

The basic principle of the proposal is illustrated in Figure 5.1. Using symmetric cryptography, the integrity of the message will be protected. To verify the authentication of the original sender, the publication of the used key will be signed using asymmetric cryptography. The basic principle may be sketched as follows, a detailed description on how to perform these steps is given in the following sections:

- Before the flight, the to-be-expected flight schedule (and a security margin afterwards) will be split in time intervals of equal duration. For every interval, a secret key will be assigned beforehand. The keys are created using a key chain.
- For every position report to send out in interval i, the aircraft A calculates a CMAC using the AES algorithm and a random, not yet published key: $C(M_j, K_i)$. At the time of sending the report, K_i is unknown to anyone but A. The CMAC of length T_{len} will be transmitted along the original ADS-B message via a second-channel as discussed in Chapter 4.
- As soon as second party B receives the report, it will be processed like a “normal” position report, but marked as untrusted. Critical actions of the aircraft should never be a reaction to untrusted messages
- To every other message sent within interval i, a CMAC, also based on K_i, is attached

- At the end of interval i, the secret key K_i is published by A. The publication of the key will be signed using a private key. Aircraft B can now verify every message send out in the interval i and mark them as "trusted". If needed, actions are to be executed.
- For the next interval, A uses K_{i+1} to calculate the CMACs. Every following position report will be protected via CMAC based on K_{i+1} till the end of interval $i+1$. Messages protected with K_i will, as soon as A publishes the key, no longer be accepted.

A brief visual presentation of this overview is presented in Figure 5.1. In this example, five messages M_j are to be sent out during the interval i. At every timestamp, denoted as x ins this example, the message M_j and the key K_i corresponding to interval i is used to calculate a CMAC to allow a verification of the message. After all messages have been sent and the interval is over, the k_i will be published. To allow an authentication of the author, the key, and the corresponding timestamp it is sent out, are signed using the aircraft's private key $-K$. In the next interval, the next key of the key chain k_{i+1} is used to create CMACs.

5.3.2. Sender Setup before Flight

Before the flight, the to-be-expected flight schedule needs to be split up in intervals of equal duration. Since changes in or delay of the schedule always has to be expected, a security margin of up to several hours, depending on the type of flight and aircraft, should be respected. The interval schedule shall be sustained for this margin as well, ending up in l intervals in total. Comparable to the basic TESLA protocol, all messages sent during the interval will be protected using the same key. At the end of the interval, the key will be published. As will be discussed in Section 7.1.1, an interval duration of 30 seconds each is proposed.

By planning all key intervals ahead, a lot of computational effort can be saved during flight. Since neither the symmetrical keys used for calculating the CMAC, nor the intervals and timestamps corresponding to the keys depend on real-time data that is created during the flight, keys and their signatures can be precalculated.

To set up the symmetrical keys, a key chain shall be used to offer a protection against message loss (as it will be described in Section 5.4). A random, secure key K_l of 128 bits needs to be created. This key will be the initial and therefore last element of the key chain. It will be used as symmetric key in the last interval of the flight. Using a one-way function based on SHA-256, as it is described in Section 2.6, a separate key will be calculated for every single interval based on the initial key. The key K_l will be used in the l-th interval, K_{l-1} in the $(l-1)$th interval, and so on. The process on how to create (and use) the key chain is presented in Figure 5.2.

To provide a possibility to verify the sender's authentication, the publication of the keys will be signed. Using the ECDSA algorithm, described in Section 2.5.4, and a secret, asymmetric private key, denoted as $-K$, corresponding to the flight number the key was created for, a digital signature will be calculated for every key. To provide an adequate

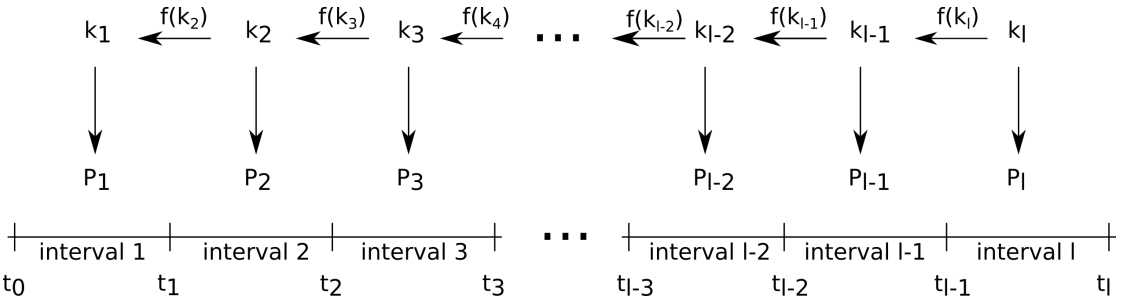

Figure 5.2.: The key chain is created by applying a one-way function f for l-times to the initial random, symmetric key. For every key of the chain, a *key-packet* P is build

level of security using ECDSA, a private/public key pair -K/+K of at least 250 bits each is required. Using 256 bits, a signature of 512 bits is created, providing a symmetric encryption security level equivalent of 128 bits. To protect the security model against replay attacks, the timestamp corresponding to the end of the interval the respective key is used in (and therefore the scheduled publication time of the key), will also be included in the signature. More information about the need to add a timestamp to the key will be presented in Section 7.2.3). In the end, for every interval i of the schedule, a *key-packet* including these three pieces of information is created:

$$P_i = \{K_i, t_i, ECDSA_{-K}(K_i, t_i)\} \tag{5.1}$$

The timestamp shall be transmitted as traditional unix-time. Using 32 bits, this timestamp allows a second-precise timing synchronous to the Coordinated Universal Time UTC. In sum, the key-packet will consist of

$$\begin{aligned} &\overbrace{128\ bits}^{\text{symmetric key}} + \overbrace{32\ bits}^{\text{timestamp}} + \overbrace{512\ bits}^{\text{signature}} \\ &= 672\ bits. \end{aligned} \tag{5.2}$$

In case the flight schedule is delayed for a longer time than expected and planned in, the signing procedure has to be started anew, since timestamps will have to be changed. The calculated key chain will be shifted to the new intervals.

5.3.3. Message Creation during Flight

When sending out a report in interval i, the transponder will create the ADS-B message itself as usual. Now, additional to the message, a CMAC will be calculated and attached to allow receivers to verify the integrity of the received message. As inputs, the key K_i corresponding to the current interval, the whole ADS-B message and the desired bit-length T_{len} of the CMAC are fed to the CMAC algorithm described in Section 2.5.3. The original CMAC given out by the algorithm will have a bit size of 128 bit, corresponding to the block size of the AES algorithm. Since a maximum of 112 additional bit,

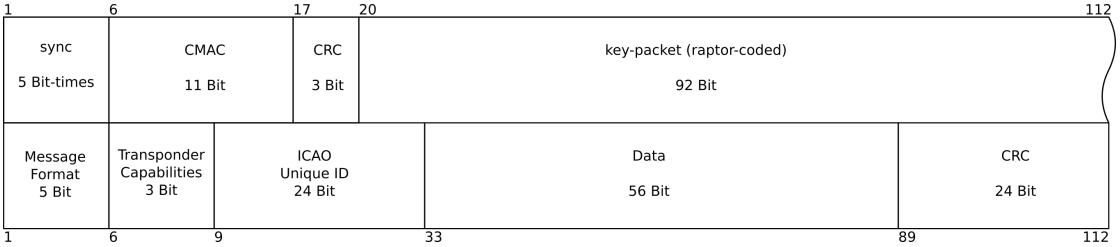

Figure 5.3.: Packet format of the adapted TESLA message. The original ADS-B message will be modulated by classical pulse position modulation. The additional information on top of the message shall be transmitted simultaneously using second-channel communication

corresponding to the original length of the ADS-B message, is expected using second-channel communication, the CMAC has to be shortened. By specifying T_{len}, the MAC will be truncated to the given amount of bits. The used bit size will directly influence the vulnerability to guessing attacks against the system and will be discussed in Section 7.1.3. As a compromise, T_{len} shall be restricted to 11 bits. To protect the CMAC itself against interference, a CRC checksum of 3 bits will be attached to the CMAC. Including the possibly needed synchronization phase, up to this point, 19 bit of the additional transmission are needed for synchronization and sending the CMAC. In case the additional second-channel transmission will not allow sending additional information of this size, the length of the CMAC may be reduced to an arbitrary length. Even a very short CMAC of only 5 bits may be possible when taking additional measures, as will be discussed in Chapter 7.

Since the key will be sent out at the end of the interval, and the timestamp sent along with the key will determine the last point in time a message using this key will be accepted, messages sent at the end of an interval are endangered to be rejected by the receiver when being delayed and receipted after this point in time. Implying a transmission range of up the 370 km for ADS-B, and a signal velocity correlating to the speed of light (299,792,458 m/s), a message suffers from a propagation delay of

$$d_{prop} = \frac{370\ km}{299,792,458\ m/s} = 0.001234\ s = 1.234\ ms \tag{5.3}$$

Additional to the propagation delay, a transmission delay of

$$d_{trans} = \frac{112\ bit}{1\ Mbit/s} = 0.112\ ms \tag{5.4}$$

is to be expected. Additional delays caused by processing and queuing need to be considered. These delays highly depend on the current workload of the transponders. In total, a timing-margin of 500 ms, corresponding to one position report sent out, shall be granted. Therefore, the last message using the interval key K_i shall be sent out at least 500 ms before the actual end of the interval. Messages created afterwards shall be

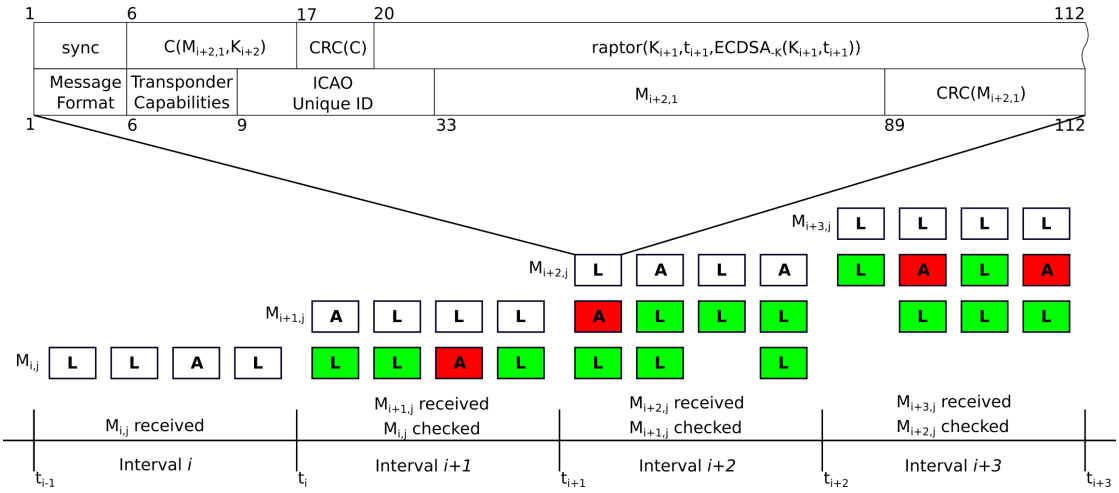

Figure 5.4.: Schematic sketch of the adapted TESLA protocol in action. A valid aircraft sends out legit messages denoted by L. An attacker injects fraud messages on the link denoted by A. Both are marked *untrusted* upon receiving. As soon as the key of the previous interval is received, all messages in this interval will be checked. Valid messages will be marked as *trusted* (green), fraud messages will be discarded (red).

using the key of the next interval. At the end of the interval i, the key-packet P_i is to be published. To avoid additional transmissions to not crowd the band any further, the key-packet shall be sent along with the CMAC and CRC of the following messages in the next interval using the second-channel communication. Since only 93 additional free bits are available per message, and the key-packet itself consists of 672 bits, it needs to be split up and send along several messages. Using 93 bits per report, at least 8 messages are needed to transmit the complete key-packet. Being split up, the key-packet is highly vulnerable for packet loss. Therefore, this transmission shall be protected by encoding it via Raptor Codes as presented in Section 2.7.

Using Raptor Codes, an theoretically infinite stream will be used to encode the key-packet. This stream will be sent along partitioned over all messages sent in the following interval. Considering position reports alone, during an interval of 30 seconds, 60 reports are sent. If each packet carries 93 bits of the Raptor-encoded stream, a stream of

$$93\ bits \times 60 = 5580\ bits \tag{5.5}$$

is sent in total per interval. As described, since Raptor Codes are considered near optimal, a correct reception of 674 bits out of these 5580 bits stream are sufficient to decode the message with a probability of more than 99.9999%.

The resulting packet format is illustrated in Figure 5.3. A sketch of the packet format "in action" is presented in Figure 5.4.

In case the secondary communication does not allow additional bit transmissions this

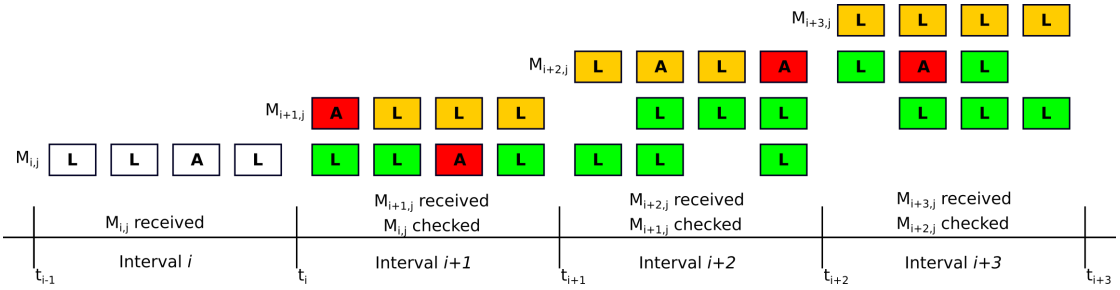

Figure 5.5.: Using sanity checks, newly incoming position reports can be pre-verified as long as the corresponding key is not disclosed. White boxes illustrate untrusted messages. Yellow boxes are pre-verified successfully. Trusted messages are presented in green, while red messages are rejected. $M_{i+2,2}$ represents a fraud message that still fulfills the limitations of the sanity check

long, the signed key shall be sent out using different procedures, that will be discussed in Section 7.1.2.

5.3.4. Message Validation by the Receiver

Upon receiving, the CMAC of the message can not be checked since the key has not been published yet. Therefore, at first the report will be processed like normal, but marked as *untrusted*. Critical actions of the aircraft, like avoidance maneuvers, should never be based on untrusted messages as long as a collision is not imminent or an air traffic controller states otherwise. The as untrusted marked messages will be buffered until the corresponding key is received.

Since the key, timestamp and signature of the previous interval are send along encoded in Raptor Codes with every message, the last 93 bits of the second-channel transmission have to be buffered separately. As soon as enough bits are collected to decode the key-packet completely, the key can be retrieved. Using the public key, denoted as $+K$, assigned to the aircrafts call sign, the signature and therefore the symmetric key and timestamp can be checked for authentication and integrity. The timestamp will dictate which messages are to be checked using the retrieved key (less the 0.5 s timing margin addressed before). Since the symmetric key K_i is now authenticated, the receiver can assume, that all messages received in the corresponding time interval that can be verified with this key were originally sent by the authenticated sender (in fact, there is still the possibility of a collision, that will be discussed in Section 7.1.3).

Using K_i, the receiver calculates the CMACs for all messages received in the corresponding interval. If the calculated CMAC matches the CMAC sent along with the message, the integrity of the message is verified (see Figure 5.4). As soon as one interval of messages is verified regarding integrity and authentication, sanity checks can be used to pre-verify position reports of the following interval. Since the position of the observed aircraft can only change in very limited ways, reports that do not match these limitations

can be sorted out before being processed any further. Still, pre-verified messages may not be used as a base to execute critical actions, since forged messages fulfilling these public known limitations can still be injected. This additional type of verification is illustrated in Figure 5.5. The concept of sanity checks is further explained in Section 5.5. A block diagram, summarizing the process of message creation and reception will be presented in Figure 6.1.

5.4. Usage of Key Chains to counter Message Loss

Key chains shall be used to provide protection against message loss. Assuming, the transmission of the key for a whole interval gets lost, meaning despite the usage of Raptor Codes the key can not be decoded correctly. A key based on a key chain could be used to retrieve the lost keys on the receivers side. As long as the next key is received correctly, the key belonging to the previous interval can be retrieved by applying the one-way function. If more keys in a series get lost, the keys can still be retrieved. Using this key chain, the features of TESLA will come back to the concept: if a key may be decoded correctly, but not the corresponding signature, the key will still provide authentication if it matches the key chain of an already authenticated key. Since the problem of the possible missing commitment when starting to receive a message in the middle of a key chain is to be omitted, and there should be no additional computational load to the clients if not absolutely necessary, the signature will still be sent along in every interval, making every key publication a new commitment.

5.5. Sanity Checks

A very simple way to increase the certainty for being secure could be based on sanity checks. As soon as at least one chain of position reports is verified by using the signed (and therefore authenticated) key to check the CMACs, the possibilities to claim current positions in the following interval are very limited. Implying a top flight velocity of an modern airliner of 1000 km/h (277.8 m/s) maximum, which is more than any modern airliner may reach, and a duration between two position reports of 600 ms at most, the euclidean distance between the two position reports can not be higher than 150 m. Considering a key publishing interval of $D_{interval} = 30\ s$, the resulting radius for this *area of uncertainty* would have a maximum of 8.33 km. A simplified illustration of this are is given in Figure 5.6.

Therefore, the euclidean distance between the last position report verified and the last position report of the chain may differ at most by this maximum, allowing an error between the last claimed position and the actual current position of 16.66 km at most (implying the aircraft would be able to fulfill a direct turn by 180°). Only positions reports within this area of uncertainty may be considered possible, restricting the potential positions in not yet verified (or negated) messages very strict. In case of missing key publications (because of message loss or jamming), this area of uncertainty may be extended in ever interval. Considering the turning radius of modern cruisers

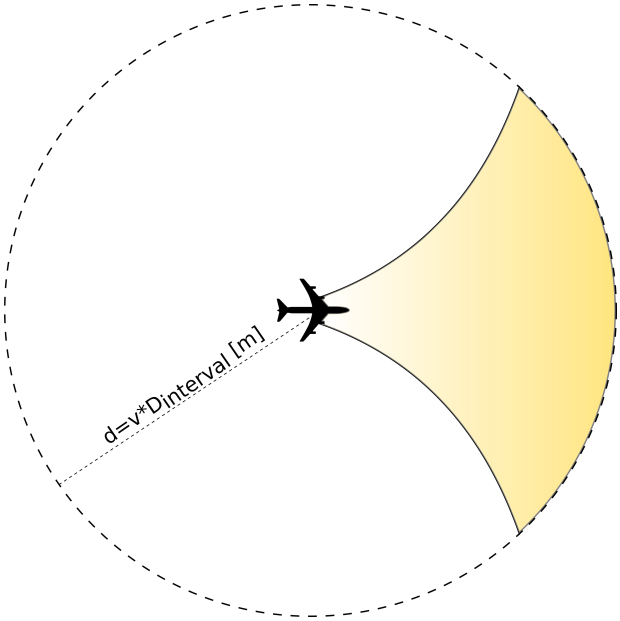

Figure 5.6.: Simplified illustration of the area of uncertainty. Based on the turn radius and the velocity of the aircraft, valid position reports following a verified report may only be sent from positions included in the yellow area

flying with velocities that high, the potential area of uncertainty may be restricted way further (in the end allowing only a small cone for potential newly claimed positions).

5.6. Public Key Infrastructure

To allow the verification of the senders authentication via asymmetric cryptography, a trusted public key infrastructure has to be established. Since call signs have to be given out and verified by the ICAO, this organization would fit in the role of maintaining the PKI. As soon as a call sign is given out, an asymmetric key pair shall be created. The private key will be given to the new owner of the call sign, the public key will be stored, matched to the call sign, in a database matching call signs to the corresponding public keys. Since one specific key is assigned to every call sign, aircraft allowed to use more than one call sign have to change the private key accordingly. The database shall be publicly available via suitable channels. Since no secret information will be stored, it could be even publicly available in the internet. In case of a stolen keypair, the key may be revoked and a new one will be published. Since the ICAO is already in charge for the ICAO public key directory (ICAO PKD, [25]), a directory to manage all keys and certificates to provide services to authenticate ePassports, the organization seems suitable for this job.

Yet to be discussed is the question, how air traffic controllers or aircraft get access to

the public keys while using the protocol to verify the signatures send along the symmetric keys. In the case of aircraft receiving reports via *ADS-B in* (if equipped), an integrity verification is possible without access to the database by validating the CMAC. But to check the authentication of the reports, the database is needed.

5.6.1. Access via the Internet

Every ATC tower should have a hold of an up-to-date dump of this database at any time. Since every tower should be connected to the Internet this is easily doable. A copy of the database shall be stored in the computer systems of all ATC towers and be synchronized on a regular basis. A push-notification system would allow to notice key revocations in near-realtime. Another problem is the distribution to the aircraft. Nowadays, modern commercial airliners and business jets are usually equipped with an Internet connection [6]. These connections are usually based on satellite constellations to support cockpit communication. Using modern technologies, transmission rates even suffice to allow passengers to use the Internet during flight for work or entertainment. This Internet connection could be used to check up public keys corresponding to messages received online as soon they are needed. But on the downside, not every plane is equipped with an Internet connection like this. Especially small, private planes may be excluded from this kind of key distribution. Since those planes usually fly only on low altitudes over land, a backfitting of an Internet connection via mobile Internet could be considered. Further research and analysis is required to check the availability of Internet connections of all ADS-B users and the possibilities of backfitting connections where they are missing.

5.6.2. Live Verification via ATC Towers

As a second option, an additional communication to the next reachable ATC tower could be used to check up the public key belonging to a received signature as an replacement of the missing Internet connection. While allowing an active verification without storing additional information, this solution would only work in areas with an ATC tower in range. Additionally, more transmission would be required to communicate with the tower, adding more data to the band, what is an contradiction to the effort to not crowd the band any further.

5.6.3. Database Dumps in every Aircraft

If an active Internet connection for all participating aircraft will work out to be impossible, it could be considered storing a dump of the database offline along to the transponder in every aircraft, tracking all active call signs. Before every flight, the database would have to be synchronized to the most recent status stored at the ICAO to know about all key revocations and new call signs. This dump would allow a fast and independent verification of all signatures received. But, depending on the amount of active call signs, this dump could be quite big. Since its establishment, the OpenSky association tracked nearly 350,000 unique aircraft/call signs. Considering the length of the call sign of 24 bit

and the key length of 256 bit alone, a full dump would require

$$(24\ bit + 256\ bit) \times 350,000 = 12.25\ \text{MByte} \tag{5.6}$$

of data. An overhead for the database structure has to be considered additionally. This amount of data seems reasonable to be stored within every aircraft. A more sophisticated approach would only store the aircraft to be expected at this time of flight and always-present emergency call signs in the local dump, depending on up-to-date flight plans. But as mentioned, maintaining the timeliness of the database will be crucial. Methods have to be found to ensure a synchronization to the ICAOs database before every flight. Time critical changes, like key revocations of stolen keys noticed during flight, would not be identified by the aircraft without an additional emergency-protocol.

5.6.4. Certificates

Instead of managing the verification directly via the public keys of the aircraft, a certificate system comparable to the one used in the Transport Layer Security (TLS) standard could be established. A root certificate, created by the ICAO, could be used to sign all of the aircraft certificates. As long as the root certificate is present in every transponder, messages signed by the aircraft could be validated using the root certificate. This solution would allow a verification using only very little additional space and computation and could be used without an active connection to the ICAO. On the downside, additional communication overhead when sending out the ADS-B messages would be required, since now a certificate and the aircrafts public key would have to be send along with every message. Considering the rate of data compared to redundancy bits in the second-channel communication, additional payload bits could be allowed to be sent along. But since this would reduce the rate, the reliability of the raptor encoded stream would be lowered. Additionally, certificates would have to be renewed quite often to guarantee their timeliness, requiring additional communication and updates before every flight. Again, emergency changes can not be detected.

6. Implementation

To provide a short proof of concept, an example implementation was build to test the feasibility of the proposal. Since the project to determine a way on how to send the additional information is still a work in progress, it is not possible to provide a real world implementation at this time. And since a lot of parameters are being researched, the full scope of operation can not be implemented at this point. An exemplary design of the blocks needed for a full implementation is presented in Figure 6.1. However, using real data based on data mining by OpenSky, the concepts can be tested on realistic data and therefore be analyzed to figure out parameters like an adequate number of bits that are needed to obtain a secure integrity verification.

The scope of the example implementation focuses on researching suitable choices for parameters used in a secure CMAC generation and validation. By analyzing the influences of different keys and MAC sizes a better insight shall be won regarding the following issues:

- What will be a suitable bit size for CMACs?
- How will the truncation of the MAC influence collisions and false positives when checking messages that are not verified with the right key?
- Are keys that produce a collision more prone to producing more collisions in sum than other keys?
- Will the truncation of the MAC or repeating patterns in the message lead to patterns in the MAC that will allow a receiver to infer back to the key?

By truncating the MAC to only a few bits, collisions are unavoidable. By being able to reliably produce collisions of the MAC, an attacker would be able to forge valid messages. But the to be expected chance to produce a collision may be easily determined and the risk that will be accepted needs to be discussed. The experiments are used to observe of the to-be-expected chance will hold for different MAC sizes or if the CMAC algorithm or the truncation will increase the chance for forging messages by producing collisions. Keys that produce large collisions are analyzed for the capability of producing collisions reliably or only by chance.

6.1. Accessing OpenSky data

As presented in Chapter 2, the OpenSky association offers the possibility to download real world ADS-B data in bulk. The data is provided in the Apache Avro format [51].

Apache Avro is a data serializing system, originally developed to provide a serialization format for persistent data and a wire format for communication in Apache Hadoop. Data types and protocols are defined using the JavaScript Object Notation (JSON). The main advantage offered by Avro is the compact binary format the data is serialized in.

The serialized ADS-B sensor data gathered by OpenSky is stored via Avro using the schema as presented in the following Listing 1 [59]:

Listing 1 JSON-described data schema as it is stored via Avro in OpenSky

```
{
"name": "ModeSEncodedMessage",
"type": "record",
"namespace": "org.opensky.avro.v2",
"fields": [
        {"name": "sensorType",         "type": "string"},
        {"name": "sensorLatitude",        "type": ["double","null"]},
        {"name": "sensorLongitude",        "type": ["double","null"]},
        {"name": "sensorAltitude",        "type": ["double","null"]},
        {"name": "timeAtServer",        "type": "double"},
        {"name": "timeAtSensor",        "type": ["double","null"]},
        {"name": "timestamp",        "type": ["double","null"]},
        {"name": "rawMessage",        "type": "string"},
        {"name": "sensorSerialNumber",        "type": "int"},
        {"name": "RSSIPacket",        "type": ["double","null"]},
        {"name": "RSSIPreamble",        "type": ["double","null"]},
        {"name": "SNR",        "type": ["double","null"]},
        {"name": "confidence",        "type": ["double","null"]},        ]
}
```

The entry of interest at this point is the field “rawMessage”. Here, the actual bitstream the transponder received is stored, encoded in hexadecimal.

To get access to the information, the OpenSky association published a set of example tools written in Java to analyze the serialized data [52]. The example tools offer basic methods for summarizing information of the downloaded bulk data or to filter messages that were sent withing a certain area. For further use, the tools offer additional methods to decode messages and output them as Keyhole Markup Language (KML) files, that can be layered upon maps using programs like Google Earth, or to store all information included in an SQLite database. Helper functions allow sorting and splitting of the bulk data.

The example avro file used for these experiments consists of 215,976,023 entries gathered in a time window of 50 minutes in January 2017. Out of 1,000,000 entries, approximately 71,000 entries were postion reports.

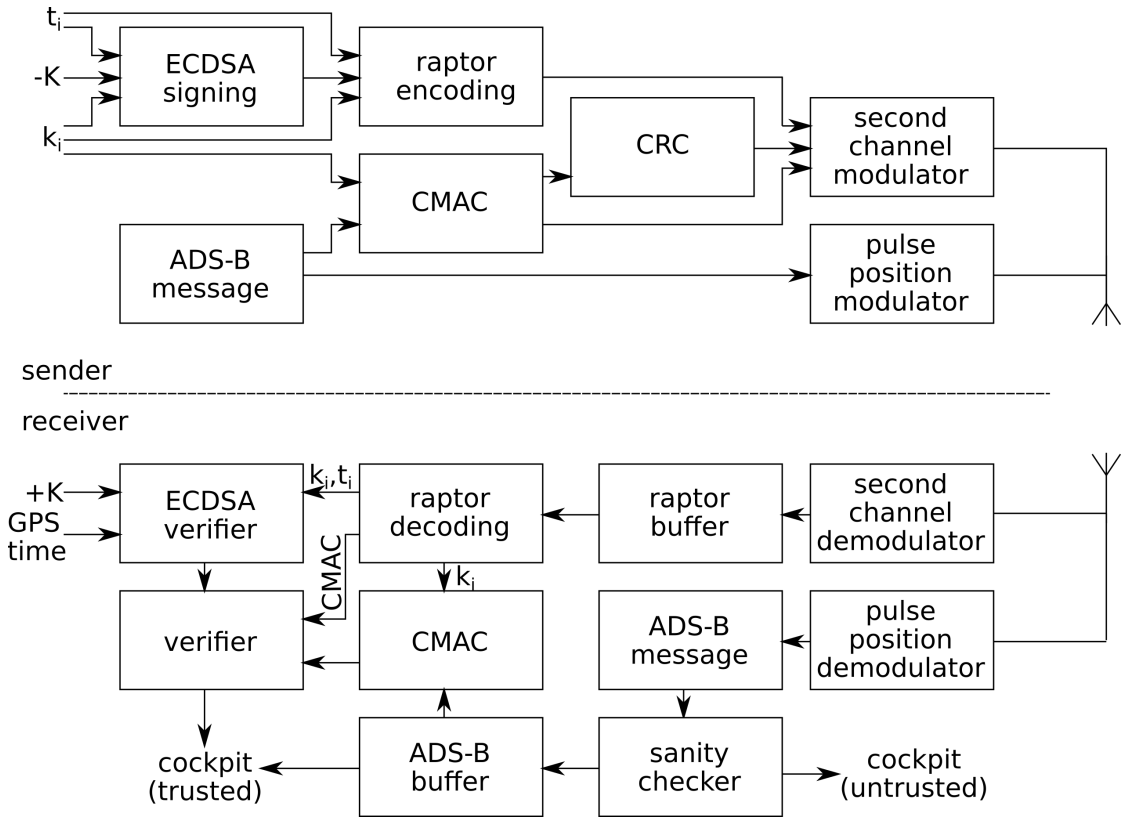

Figure 6.1.: Exemplary design blocks needed for an implementation of both sender and receiver. Since most of the parameters of the design are still to be researched, the example implementation focuses on the blocks for creating *CMAC* and the *verifier* to get a better picture of the required input parameters

6.2. The Bouncy Castly Crypto API

To provide cryptographic functions, like the CMAC operation, symmetric key creation or one way functions, the *Bouncy Castly crypto API* [50] was used. The APIs are maintained by the *Legion of Bouncy Castle Inc.* and provided for free to use and open source under the MIT license, offering APIs for Java and C# applications and a set of additional tools. Additionally, APIs are available in a FIPS hardened version. The FIPS defines Security requirements for cryptographic modules in the USA.

Unfortunately, beside a set of testing classes and coding examples, very little documentation is given to work with Bouncy Castle. At this point, some of the authors provide books explaining the basic functions [24].

6.3. Altered Sample Reader

The example implementation to create key(chain)s and calculate the CMACs is directly built on top of the OskySampleReader class presented on the OpenSky sample tools, originally authored by Markus Fuchs[1].

Since the data of interest for this proposal is the position reports, the provided ADS-B message decoder was rewritten to filter exactly this type of message (downlink format 17 - ADSB_AIRBORN_POSITION). Other packet formats are ignored at this point and therefore just skipped when crawling to the messages.

For every position report filtered out, a CMAC was created using the Bouncy Castle Crypto API. For this, a set of secret keys or key chains were randomly created and assigned to every ICAO call sign included in the example data. To build up a comparison, the aircraft that sent out the most captured position reports within the examined time window (further denoted as reference ICAO) was associated a fixed key/key chain (reference key). This key was later used to test it against all other MACs regarding collisions and false positives. The created data was gathered in a CSV-file (comma seperated value). For every report, the associated call sign, the secret key, the report itself (encoded in hexadecimal and binary format) and the created CMAC (in hexadecimal and binary format) was stored.

By using different ways to feed the ADS-B messages to the CMAC algorithm the consequence of repeating patterns in the message was studied. First, the ADS-B messages were fed in as a whole, including message format, transponder capabilities and ICAO call sign, that are equal for all messages from the same transponder. Next, only the payload of the ADS-B message (meaning the message itself and the CRC) were used, forcing the CMAC algorithm to apply a lot of padding at the end of the message. Corresponding, a repeating payload and a payload concatenated to random bits was used (the latter will not really represent a possibility in real-world, since the random bits would be also needed at the receivers side to verify the MAC).

A Java source code example for the implementation can be found in Appendix A.1. For the evaluation, the data was not restricted to a specific region. Since the main goal of the implementation was to find potential weaknesses in the creation of the CMAC and to determine a suitable bitsize for the authentication codes, the more data was available to examine, the better. It has to be noted, that in reality, based on the transmission range of an ADS-B message and the required separation between the aircraft, never this many messages would have to be processed at the same time. A short example of the data is given in Table 6.1.

6.4. Verifier

To further analyze the data, a verification module was written. For every message sent by an aircraft with another call sign than the reference ICAO, the CMAC is calculated using the reference key. This newly calculated CMAC is tested against the CMAC “sent

[1] fuchs@sero-systems.de

along" with the message using the original key belonging to the original sender of the message. The two MACs are compared for equality. If a bit-wise collision is found, it is counted how many following bits collide. All results are again stored in a CSV-file, listing the message, the original MAC sent along, the MAC calculated with the reference key and tests for several bitlength for equality. If an collision was found, this will be marked as a false positive for the related bitlength.

Messages/keys resulting in the largest collision were further analyzed. For the keys producing the longest collisions, it was observed if these keys are more prone to producing collisions than other keys. The produced MACs were observed for repeating patterns that could lead back to the key. Again, an example of the data is given in Table 6.2. A source code example of the verifier can be found in Appendix A.2.

icao24	key	msg_hex	mac_hex
394c0f	75a4dc0bc01925d6c73cf87a9ce476f5	8f394c0f583d0062b2842ef426bc	b912bf2eb2e798643c062cfa00ab4c32
4841db	cc1f767ce810a58e7cac357b5cfedda4	8d4841db60c900a81871a82b43f1	d894872062ecd1a59f44c3cacf38f7c0
3423c4	38cda177cc33425353596bc5708ac82f	8d3423c45817d67a1d281a000000	eb02ae80c1192518b90a835eb52aab47
780ab4	575c0783090b29284b72a001a1896f5e	8d780ab458af768e7ced0b6ad6ef	d6169c62218dff74031404edc9072278
a757ae	0d0f8b3c2d1c2370f8133b479568e16c	8da757ae58af96f2bfcfdd7da34d	8293caab292e1e33ee3d558e11e18936
a8c3ea	07ceff1d5183c6e65dea10f1e91c6036	8da8c3ea6075282b9f29f74eec09	d0339209ce2244e5984e4658d3e54627
a68915	9635506d21788624b4053c8273aae6b8	8da68915906dc02aeda7b3d25110	ec6bb58d74fa0b7a791d7e285c275068
71c044	8ddbfd3c5da6f446f73c0675d6e36295	8d71c04458a103945f2066d130ed	80265116c41349a554544abb6b5a00f5
86e888	ca0c051b1c1b2f7b1f6723bda852f8a8	8d86e88858c372c04f8f4ee71784	84c6932deb2957bd970cf87fc980b568
8962d1	964c6587a1221fce9af23c3c2d3ee5eb	8d8962d158a586cb4e94d4044af0	c5b94b83b2e55a0855ff83135fa0cb17
880051	02bfe05d80166f6481f37a18d807be72	8d88005158a58313a9a168dc122a	a4c28ccdc238c03efbc43a2747213dc2
780a58	6fc172dc2b640a6ae206515f4cdacc63	8d780a5858cd83dd42e4d01ae51c	4e2500311d7d32b1ca30271bc1346672
702039	480a0806f6c4d864914d00e15ea8fcd4	8d702039500fc003571d3dd0fc54	c1aca3f3126de648ce9e98bdda17fa7f
70c0a8	9059a84cb6bf1e08fb662c87cc66cf95	8d70c0a858af82fc02f4b399c041	9ac335dffb778afd01d05e12423ef37c
800bd7	9154f869f8316485008a95dee98111a0	8d800bd758b976b8570db3c15313	82d5b2b51512e5333610dbadb26cd8e8
44a832	0b13132389b6803bf7884edc7e7ce052	8d44a83258c90791fb09ab000000	c4b7fc42c84daa3fa4f2eb1a0d95f5c9
aa8b40	e12d350f0c5a73e9fd709b4f69dd2e6b	8daa8b40581b27cb856a54200f88	f889dc78d4b63971918b0a7cc5d702e9
a18dd3	b1c1df307aa8cb9442c7a5b3aef9d3b5	8da18dd358d304030968ede12610	f0636c695ac474e49618248049696ab9
abe101	152335eee5cef2ce98934479caef84b2	8dabe10158afa7af1d5e09c4d2bd	f77768c08e1b920aa3348254b229aa3d
ab7a21	bdb0775395ffb2f329da03c9ffca6ccd	8dab7a21589b87531d8d77522966	395a50338a783a5a5b1a0256c1048b87
a7b8a1	2b360516982593791d2967abe5fdfa58	8da7b8a158ab0779f3ce84182fdd	f87bb4ebc49e64f4704febb417a1ffb4
ab6cc9	29a02ba9e4e86086ea919f8ae42112ec	8dab6cc958af810295da4c8a2b93	5470310cecd4ecfe7c1e39acb86fda5f
abe8b6	c37156a19eb842d9e4e54826557db86e	8dabe8b6588bb7d5d3a306b0f94b	b3bcbcec218619192e8601871ff1cb99
342141	6223449c00898621ffc079c04cbcebb7	8d34214158c383693fd5728991c0	7be5066f50b3ce31ed0e10e8bbabe839

Table 6.1.: Short example of data generated by the CMAC generator. For every position report, a CMAC was calculated based on the key assigned to this specific ICAO call sign

icao24	msg_hex	mac_msg	mac_calc'd	#
394c0f	8f394c0f583d0062b2842ef426bc	b912bf2eb2e798643c062cfa00ab4c32	3dfab3673e4b7a71cb3e740177b4d863	0
4841db	8d4841db60c900a81871a82b43f1	d894872062ecd1a59f44c3cacf38f7c0	008e83304b3960288dc9b4f0ba56cd6f	1
3423c4	8d3423c45817d67a1d281a000000	eb02ae80c1192518b90a835eb52aab47	5c6dfc57c3f78c566c4b86b19b9772ac	0
780ab4	8d780ab458af768e7ced0b6ad6ef	d6169c62218dff74031404edc9072278	6b7f6dcfcbb0e90cc62793d0b73af957	0
a757ae	8da757ae58af96f2bfcfdd7da34d	8293caab292e1e33ee3d558e11e18936	887b83fb89ce0019de6d424cec87d88a	4
a8c3ea	8da8c3ea6075282b9f29f74eec09	d0339209ce2244e5984e4658d3e54627	9d4226b9fb8757747643177d25b65beb	1
a68915	8da68915906dc02aeda7b3d25110	ec6bb58d74fa0b7a791d7e285c275068	ecf030bb7302d01191f1089acce967a3	8
71c044	8d71c04458a103945f2066d130ed	80265116c41349a554544abb6b5a00f5	bd977f7cb1a4ee7525e371da239a3542	2
86e888	8d86e88858c372c04f8f4ee71784	84c6932deb2957bd970cf87fc980b568	fdfdbcbde533adf5c00792f11bd67e89	1
8962d1	8d8962d158a586cb4e94d4044af0	c5b94b83b2e55a0855ff83135fa0cb17	5045ff7a928330c329ab0c33062c8410	0
880051	8d88005158a58313a9a168dc122a	a4c28ccdc238c03efbc43a2747213dc2	46d857cd667c05bc77fd99e77784f567	0
780a58	8d780a5858cd83dd42e4d01ae51c	4e2500311d7d32b1ca30271bc1346672	fbe86057cbea4cfb6fff037d7c206399	0
702039	8d702039500fc003571d3dd0fc54	c1aca3f3126de648ce9e98bdda17fa7f	364afa4db3122dcc2a4877c75a600ce3	0
70c0a8	8d70c0a858af82fc02f4b399c041	9ac335dffb778afd01d05e12423ef37c	9d5cce1b5215d62eaae96a5405236763	5
800bd7	8d800bd758b976b8570db3c15313	82d5b2b51512e5333610dbadb26cd8e8	aa8cc9cd59606ae719e5da3780a05f13	2
44a832	8d44a83258c90791fb09ab000000	c4b7fc42c84daa3fa4f2eb1a0d95f5c9	e25973e876b1f7095065275682782e81	2
aa8b40	8daa8b40581b27cb856a54200f88	f889dc78d4b63971918b0a7cc5d702e9	bb66a6d409bff200cfd9f66c210bb73d	1
a18dd3	8da18dd358d304030968ede12610	f0636c695ac474e49618248049696ab9	81b10da6a05d2fee27c5b2eb5d35b889	1
abe101	8dabe10158afa7af1d5e09c4d2bd	f77768c08e1b920aa3348254b229aa3d	0b208479b762ca937fe7736a29039ab8	0
ab7a21	8dab7a21589b87531d8d77522966	395a50338a783a5a5b1a0256c1048b87	a7fe9ef76025aab038b83bd346a80759	0
a7b8a1	8da7b8a158ab0779f3ce84182fdd	f87bb4ebc49e64f4704febb417a1ffb4	81107918cecf581ea2347f1fc0a2162a	1
ab6cc9	8dab6cc958af810295da4c8a2b93	5470310cecd4ecfe7c1e39acb86fda5f	9411294b76b7a12842ed5fe9b3c70c04	0
abe8b6	8dabe8b6588bb7d5d3a306b0f94b	b3bcbcec218619192e8601871ff1cb99	1676895e65ea2da7170d4752709c8f9d	0
342141	8d34214158c383693fd5728991c0	7be5066f50b3ce31ed0e10e8bbabe839	bc4c93e472d3f567e1a9f07743278ce5	0

Table 6.2.: Short example of data generated by the verifier. For every message, a CMAC is calculated using the reference key (in this case set to the hex string *2b7e151628aed2a6abf7158809cf4f3c*). Bit by bit, the "send along" MAC is compared to the calculated one. The number of colliding bits is tracked in the most right column

7. Analysis and Discussion

7.1. Design Decisions and Parameters

Computational Effort and precomputed Keys

As mentioned, the computational power of transponders involved in the ADS-B ecosystem is quite low. This leads to the requirement, that transponders are not to be set under unnecessary load when using the security model.

By using the CMAC algorithm based on the well-proven AES algorithm, a possibility opens to produce reliable message authentication codes by still going easy on resources. Hash based MACs or especially asymmetric cryptographic algorithms usually require a high load that needs to be calculated based on software. AES on the contrary is nowadays implemented in hardware or hardware supported by special opcodes in most modern processors and microcontrollers. Short keys and short block sizes allow easier computations than competing algorithms for the short message sizes that are used in ADS-B. Therefore, the CMAC presents the best choice for fulfilling the requirement.

Way more complex, in a computational sense, is the calculation of the key chains and the signatures. Especially ECDSA requires very complex computations, since it is based on elliptic curve cryptography. Here, the system may profit from the fact, that the signatures are not dependent on the created real time data of the aircraft at all. Since only the secret key and the corresponding timestamp are signed, the key chains as well as the signatures can be precalculated before the flight. Maybe even modern commodity computers may be used to create all the information before storing it in the transponder. The required storage for the key-packets it quite manageable:
The longest (by the time of this thesis still planned) non-stop flight possible, in the year of 2018, would be the route Sydney-London with a time of flight of nearly 20 hours [68]. Adding the described security margin for delays, a time of 22 hours may be planned in (definite values for delays need to be researched further). Assuming an interval duration of 30 seconds, and 2 position reports to be send out in one second, up to

$$1320\ min \times 2\ \frac{intervals}{minute} = 2640\ intervals \tag{7.1}$$

need to be supplied with keys and signatures. Therefore 2640 key-packets need to be calculated before the flight. With a key length of 128 bits, a timestamp of 32 bits and a signature of 512 bits,

$$(128\ bits + 32\ bits + 512\ bits) \times 2640 = 1.774\ Mbits = 221.8\ kByte \tag{7.2}$$

of key packets need to be precomputed and stored in the transponder.

By using Raptor Codes, the reliably transmission of the key-packets shall be ensured. Raptor Codes allow an encoding and decoding with a linear computation complexity. This low complexity is bought by the loss of the optimality of the resulting encoded stream. But since this loss in optimality is very marginal, and enough bits are available to decode the stream correctly, Raptor Codes should be a great choice with respect to the computational effort needed.

7.1.1. Interval duration

Yet to be discussed is the interval duration between two key publications. In [7], two pilots were asked about the time they could wait in case of an TCAS warning based on ADS-B before they are needed to take actions. In this theoretical model, the range for ADS-B messages was restricted to only 100 km, and therefore less than a third than the theoretical possible value. Considering both aircraft moving at top speed of 250 m/s, both would approach with a summed up speed of 500 m/s in the worst case. Giving these restrictions, the time window before a collision may occur would consist of 200 seconds. The interviewed pilots claimed, that two and a half minute would be sufficient to start a collision avoidance maneuver, allowing 50 seconds to elapse for waiting for a verification. To offer an additional security margin, the interval duration could be set to the in this thesis frequently used duration of 30 seconds. The additional 20 seconds should offer a time window large enough to decode and process the key packet. But at this point, a more representative picture provided by pilots, air traffic controllers, airlines and manufacturers should be gathered to find a suitable duration.

7.1.2. Key Disclosure

As mentioned in Chapter 5, the way how to publish the symmetrical keys for creating the CMAC will highly depend on the possibilities offered by the secondary modulation. This thesis proposes to send out the key encoded in a Raptor stream using the second channel communication as presented in Section 5.3.3. But this kind of publication will only work, as long as enough bits are offered by the second channel communication. Therefore, alternatives have to be considered. This section provides three proposals on how to publish the keys:

Publication via Second-Channel Communication using Raptor Codes

Publishing the key-packets along with the original message using the second channel communication provides a good possibility to publish the keys without creating further traffic on the band. As long as the assumption can be fulfilled, that for every bit one additional bit can be sent, every message allows to carry 93 additional bits after synchronization and sending out a CMAC of a reasonable length and a corresponding CRC. Since every key-packet will have a size of 672 bits, the packets have to be split up and be sent attached to multiple messages. To ensure a reliable transmission considering the to-be-expected message loss, the key-packets shall be protected using Raptor encoding. Raptor Codes allow a reliable decoding of the message with a high certainty when

receiving at least the original amount of bits. Every additional bit may increase the probability even further. When assuming the possibility to send 5580 bits in total (as presented in Section 5.3.3), even when considering a message loss of more than 40% the successful reception of at least 672 bits should be possible with a high probability.

This kind of transmission allows the publication of the key packet without adding additional traffic to the band and ensures the successful reception with a very high probability, even when considering message loss. On the downside, the encoding will introduce additional delay to the verification, since at least 8 messages have to be received before the complete key-packet can be decoded in the best case. Considering position reports alone, 8 messages will correspond to 4 seconds delay. Additionally, the decoding and encoding introduces a linear computational effort on both sender and receiver side as well. But since this effort only needs to be done every 30 seconds, it could be accepted easily compared to the effort the CRC- and CMAC-calculation requires for every single packet.

Publication via an Extra ADS-B Message

In case the transmission via the second channel is not possible or desired, the easiest way to publish the key would be to send out an additional message. Since the message only has to be accepted by all clients participating in the security model, an ADS-B message with an invalid format, only accepted by the participants, could be send out. This way, no additional computation has to be considered to receive the key packet. Beginning with the call sign matching the aircraft, the following bits could be filled with the symmetrical key and the signature attached, resulting in a message of

$$\begin{aligned} \overbrace{5\ bits}^{\text{Message Format}} + \overbrace{3\ bits}^{\text{Transponder Capabilities}} + \overbrace{24\ bits}^{\text{Call Sign}} + \overbrace{128\ bits}^{\text{symmetric key}} + \overbrace{32\ bits}^{\text{timestamp}} + \overbrace{512\ bits}^{\text{signature}} \\ = 704\ bits. \end{aligned} \tag{7.3}$$

It has to be noted, that this message includes no error correction at this point. Given the assumption, that the usage of additional PSK modulation could offer the same amounts of bits the ADS-B message itself could offer, a message of at least 352 bit would have to be sent out when using both channels. Introducing an error detection code like CRC of at least 24 bits, as it is used by ADS-B messages, on both channels, the message comes with a minimum length of 376 bit.

Considering only the regular transmission of position reports alone, in an interval of 30 seconds 60 ADS-B messages with a bit size of 112 bits are send, summing up to a total of

$$60\ messages \times 112\ \frac{bits}{message} = 6720\ bits \tag{7.4}$$

are sent within every interval by ADS-B itself. A transmission of additional 376 bits would correspond to an overhead of 5.6%, that needs to be send additionally. When implying no additional information beside the CMAC can be sent using the second channel communication, what is the real use-case for this publication, an additional

Table 7.1.: Observed false positives for MACs of length T_{len}, that have been verified with the reference key despite being created with another key. 727,181 position reports were observed

T_{len} [bits]	5	10	15	20
colliding MACs	22582	726	37	0

message of 752 bits has to be send (this time considering a CRC of at least 48 bits, since the message itself is way longer). This message would correspond to an overhead of 11.2%, contradicting with the requirement of adding only as few additional traffic to the band as possible. Additionally, a message this long would be even more prone for interference and message loss. A single message loss may be recovered by the key chain and sanity checks, but would still open up gap of one minute of unverified messages.

Publication via a Third Channel

As discussed in Section 5.6.1, a transmission of information via the Internet needs to be investigated. If no other possibility is desirable, the key-packets itself may be published using this "third channel". But as mentioned, this method could lock out smaller and older aircraft. Additionally, the Internet would represent an additional, non-realtime error source, since its reliability can not be guaranteed. Additional mechanisms would have to be developed to ensure a secure and authentic transmission. Therefore, this method will not be recommended and only be seen as some kind of "last resort".

7.1.3. Bitsize of the CMAC

The only method to forge messages secured by the CMAC is to produce collisions (as Section 7.2.1 will explain in detail). Using the results produced by the verifier module, a further insight on false positives/collisions is won. A collision is found, if a key produces the same MAC that will be produced with another key too. Since the same MAC is calculated with both keys, both will accept the MAC as valid. Table 7.1 summarizes the amount of collisions when verifying MACs of 727,181 messages in total (in a total of 10,000,000 unique ADS-B messages). When using only very short message authentication codes of just 5 bits, for 22,582 messages a collision occurs. In fact, this is exactly the behavior to be expected. By offering only 5 bits to the MAC, there are just $2^5 = 32$ possibilities for unique MACs. When considering 727,181 messages, there is a chance of 3.125% that a collision will occur when calculation the MAC for another aircrafts message with the own key. Or, in other words, ever 32^{nd} will collide. Considering 22,582 collisions in 727,181, this equals a collision rate of 3.105%, quite near the expected outcome.

Using 11 bit for the MAC would offer a space of $2^{11} = 2048$ unique MACs. ADS-B regulations require the system to be stable for up to 1250 aircraft sending out messages at a time [4]. Since the only way to forge valid messages would be based on guessing attacks, an attacker would have to inject 2048 different messages to be ensured to have

sent out an valid one. An attacker capable injecting 2048 position reports twice a second will have to be considered a denial of service attack, making attacks on a verification system obsolete anyway. Therefore, 11 bits are considered sufficient to protect the ADS-B messages integrity. For a bitsize of 11 bit, only 343 collisions were produced in 727,181 analyzed position reports, which equals a probability of 0.05%.

Still, if needed, shorter MACs may be used if it is required by the second channel communication when considering additional measures. As explained above, for only 5 bit-MACs, the chance to produce a collision by guessing is pretty high (1 out of 32). But as will be stated in the following Section, finding one collision will not guarantee that this key is able to produce collisions for other messages. For the next message, the probability of producing a collision with the same key will again be 1 out of 32. In sum, the chance to produce a collision for both messages using the same key will be

$$1/32 \times 1/32 = (1/32)^2 \tag{7.5}$$

For every additional position report, the probability will fall further for the same factor. Therefore, it would be reasonable to introduce a security criterion at which point following position reports shall be accepted as a whole. Implying an interval duration of 30 seconds, about 60 position reports are to be sent out in this interval. As an example, the criterion could demand, that all position reports in a whole chain of corresponding position reports will need to be verified for the chain to be accepted. A chain may consist of all position reports that pass the *sanity checks* presented in Section 5.5. For all 60 Messages, the chance to produce valid collision will fall to

$$(1/32)^{60}, \tag{7.6}$$

making it very unlikely to produce a whole chain of accepted messages based on guessing a key. Considering the expected message loss or possible faulty receptions, the criterion could be set to a less strict demand of only 50%, still lowering the chance to possibly launch a successful attack based on guessing to a very low level. On the downside, a criterion like this would allow an attacker to start simple denial of service attacks: By injecting more knowingly wrong messages than the demand claims to be correct on purpose, that will still pass the sanity check, the attacker could force the receivers to reject the entire chain of reports. Therefore, as possible, this security demand shall only be used when really needed.

7.2. Security Evaluation

By providing a possibility to verify the integrity and authentication of an ADS-B message, "traditional" attacks like the *Ghost Aircraft Injection* will not be possible anymore. When using an unknown call sign or when trying to steal the identity of an existing aircraft, the attacker will not have the possibility to produce valid signatures without the secret key corresponding to this call sign. Therefore, by the end of the interval, the authentication of messages will fail and the fraud will be detected. But introducing

the proposed means to protect the messages will still open up possibilities to attack the system beside traditional identity theft attacks, that need to be discussed. Namely, especially when comparing to other TESLA-like systems, attacks based on message forgery, jamming and replay attacks need to be discussed.

7.2.1. Protection Against Message Forgery

The most basic attack to fool the security model will be based on forging a message that will be accepted as valid by the receiver. To launch a successful attack like this, the attacker has to either get hold of the original key before it is made public, find a valid way to reliably produce collisions, or by launching timing attacks that emerge from gaps in the key disclosure schedule.

Breaking the Key

Researchers like Dworking et al. [18] come to the conclusion, that as long as the cryprographic algorithm (in this case AES) holds, a shorter or cut MAC will not allow an adversary to guess or calculate the key with a higher probability than with a full MAC. For a key size of 128 bits, there will be 2^{128} possible keys. To find the key used for creating a given MAC corresponding to a given message, in average $2^{128}/2 = 2^{127}$ keys would have to be tried out. For every key, the CMAC algorithm would have to be applied to the message. If the resulting CMAC matches the one send along, the key is retrieved and therefore broken. But applying the AES algorithm for 2^{127} times in a row is not doable in a reasonable amount of time even when using modern hardware [45]. Since the secret key can not be retrieved out of the MACs, whatever length is chosen, an attacker will not have the possibility to create own, valid messages.

Message forgery using guessing attacks

The only alternative is guessing a key and hoping that this key will produce the same MAC as the original key by chance. As explained above, for shorter MACs this chance will rise, since now not only the correct key will produce a valid MAC, but also all keys that will produce collisions. But producing a collision for a known message will not automatically guarantee that the used key will also produce collisions for other messages: As long as AES works correctly, the resulting MACs for another message may be completely different compared to the MACs produced with the guessed key. This statement is confirmed by the experiments done using the example implementation: For a key that is capable of producing a collision for a given message, the chance for producing more collisions on other messages is exactly the same as for all other keys. Hence, the attacker can not verify that his key will produce a collision for other messages, forcing him to guess keys until he finds on key for every possible MAC. When now sending out all of this messages, one of them will definitely be accepted as valid.

Therefore, a protection against spamming is needed. This could be easily done by rejecting position reports by call signs if there is more than the to be expected amount of position reports per second received as it is already used to filter out multipath

effects. Additionally sanity checks could protect against implausible position reports. As explained above, for a MAC size long enough, attacks based on sending out this many messages will end up being a denial of service attack which can not stay unnoticed. Additional measures, like the fallback to PSR and traditional SSR would have to be taken, that are immune to attacks based on false message injection.

Message Forgery Using Timing Attacks

TESLA based security systems are often prone for timing attacks based on a small gap that might arise from the key disclosure schedule. After the disclosure of the secret key, a receiver close to the original transmitter could take the key as one of the first receivers, send it to another transmitter using a faster data link than ADS-B, forge messages signed with the original key and send it out to all receivers in range before the key disclosure reaches them. Those messages will still be accepted as valid. Therefore, the original TESLA protocol reserves a delay based on the transmission times and clock errors in the key disclosure schedule.

By using the proposed security model, attacks like this are not possible. By adding a timestamp to avoid replay attacks to every key disclosure, a hard time limit is set and a kind-of time synchronization is established. Since all involved parties get their timing information based on the very precise clocks of the global positioning system, clock errors should be very small since all calculations done by GPS are based on a precise timing. In publications like [36], researchers observed a clock error for GPS of less than a few milliseconds in the worst case. When using a key disclosure strategy using systematical Raptor Codes, at least two ADS-B messages have to be received before the key can be recovered, since a key of 128 bits would take two of the 93 bits windows offered by the second-channel transmission (and at least 8 messages are needed before the report can be verified). By sending out position reports twice a second, one second of delay will be needed before the key can be recovered—way more than the possible clock error. Therefore it should not be possible to retrieve the symmetric key before all receivers register the end of the interval. When using another key disclosure strategy, an artificial key disclosure delay of at least one second should be introduced to be on the safe side.

7.2.2. Protection Against Jamming

A potential attack on the system could consider precisely jamming all key publications, to make a verification of the CMACs impossible. The threat by this kind of attack may be avoided by sending the key partitioned over all position reports as presented in Chapter 5, making a jamming of just the key infeasible without jamming the whole system. In the case the key will have to be revealed in an additional message as discussed in Section 7.1.2, this protection is lost. The single transmission of the key would represent an easy target for being jammed. With the proposed usage of key chains and sanity checks, the threat based on those attacks is reduced, since at least uncertain verification is still possible even without the published keys. An attack based on jamming for a longer time span will not stay unnoticed, allowing the ATC to start countermeasures

(like passive verification, traditional verification by PSR, or finding and stopping the attacker).

7.2.3. Protection Against Replay Attacks

Keys used in the presented proposal are only valid until their publication. For every receiver in range, this would allow only very short time spans to fraud messages with the released keys.

Another problem could arise for all ATC users that are not in range of the aircraft whose identity shall be stolen. Theoretically it would be possible for an adversary to mine information sent out by an aircraft at position A. After the key is released, the adversary could build own, valid MACs based on this key to publish at the far away position B. All participants at B are not aware of the age of the key used by the adversary, given that they did not receive the key when it was published. Since the disclosure message was signed by the original sender and in fact all CMACs were created correctly using this signed key, the replay attack would be successful.

To face this problem, a timestamp is send along when publishing the key. The timestamp will be signed along with the key and is therefore protected against forgery. Forging the timestamp would render the signature useless, therefore the key would not be accepted by the receivers. Replaying the original, signed publication of the key would not allow the receivers to verify the currently received messages, since the timestamp of the publication will not match the recently passed interval. On the downside, this replay protection will come with the need to send an additional amount of 32 bits with every key disclosure. The only alternative would be to publish the key disclosure timings along with the public key. But this again would require permanent updates of the PKI and introduces a lot of additional information that needs to be transmitted and/or stored. Therefore, the timestamp seems to be the better choice.

7.2.4. Observed Security Issues by Repeating Patterns

Some cryptographic methods suffer from weaknesses, if data that is to be encrypted consists of repeating patterns that may be found in several messages that are all encrypted with the same key. In ADS-B, several of these patterns can be found. Every ADS-B position report of one specific aircraft will start with the same 5 bits to determine the message format, the same 3 bits for transponder capabilities and the following 24 bits for the ICAO call sign. The following 80 bits are arbitrary, since they are based on the actual position. With the usage of a strong, random key, subkeys and permutations, AES-CMAC is considered hardened against attacks based on repeating patterns. In [35], comparable attacks were analyzed regarding the possibility to forge valid MACs after an increasing number of messages has been received. The authors find “that multiple forgery attacks are more of a theoretical concern than a practical one” [35].

Even if very unlikely, one could still want to reduce the risk for a successful attack to be found. A possible solution would be to simply only feed the payload of the message, meaning only the position report itself to the CMAC algorithm. Since the message

would be even shorter than, and a lot of non-arbitrary bits would have to be padded to fill the blocksize. Therefore, a concatenation of repeating payloads could be a solution. During the experiments and observations described in Chapter 6, no obvious patterns or weaknesses were found regardless the way ADS-B messages were fed into the CMAC algorithm. Yet, further research may be done to fully exclude attack vectors in this topic. However, it has to be mentioned, that an attack using the repeating patterns would still have to very much lower the level of security to allow successful attacks on the system. Caused by the short interval duration before the key is released, a possible attack would have to retrieve the key or the possibility to forge messages in a very short duration, that is even for small, insecure keys hard to beat using modern hardware. Additional measures like the sanity check could even lower the possibility to forge messages successfully, since earlier messages could kind-of protect later messages. When analyzing the CMACs produced via the example implementation, no weaknesses or conspicuous patterns could be observed.

7.2.5. Evaluation of Integrity and Authentication Mechanisms

To omit unnoticed alteration of the position reports, CMACs are used to verify the integrity of the message. Based on a strong and reliable algorithm, the security of the CMACs is dependent on the possibilities that are offered by the second channel. But even for very short CMAC, a set of helper tools and restrictions would allow a secure protection of the message's integrity. Additional sanity checks could also provide an additional level of security with very little cost.

The authentication of the messages, the main goal of this proposal, is secured using strong asymmetric cryptography. It has to be mentioned, that this protection comes at high cost, since digital signatures protecting the authors authentication require a lot of additional bits to be sent and a trustworthy database to maintain the keys. Additional storage or communication overhead has to be expected. Nonetheless, as long as the PKI is uncompromised, no attacks are known that could attack the authentication provided by ECDSA signatures. The usage of key chains and sanity checks may make the authentication protection even stronger, allowing a kind-of verification of messages before a real verification based on the secret key can be performed.

It is noteworthy to mention, that the proposed security model will in fact provide a verification regarding the message's integrity and the author's authentication, but the message content itself can not be protected. Attacks can no longer be started by third parties using the right equipment, but from "enemies within". As long as an aircraft's authentication is verified, all of its messages will be accepted. Forged position reports by this actual aircraft will not be noticed to be wrong. At this place, as long as it can not be ensured that only trusted parties are part of the system (and this may be never possible when using an open, world wide net like ADS-B), additional measures based on passive data verification, as presented in Chapter 3, are still recommended.

8. Conclusion and Future Work

8.1. Conclusion

This thesis proposes a design of a model to ensure the integrity and authentication of ADS-B messages. By studying related work, the needs, requirements and possibilities on how to protect the modern air traffic control messages were identified. To introduce a security model while still ensuring the legacy support and reliability of the existing system, a *second channel communication* shall be used to add additional information to the existing signals by combining the used pulse position modulation with another kind of modulation. Different possibilities adding information by exploiting physical properties of the signal have been researched an discussed.

Based on the possibilities to be expected by the second channel communication, a security model to protect the ADS-B data was proposed. The model focuses on providing a high level of security while not altering or disturbing existing ADS-B transmission. Using cipher based message authentication codes and digital signatures based on elliptic curve cryptography, the integrity and authentication of the messages shall be secured. The design was chosen in a way, that all processes and transmissions ensuring the security of the messages are reliable while requiring as little computational effort as possible.

A step-by-step solution was presented on how to manage cryptographic keys, how to protect ADS-B messages during flight and how to verify the data at the receiver's side. By implying a successful transmission of a reasonable amount of additional data, parameters have been determined and discussed to fulfill the elaborated requirements. An example implementation was built to study the requirements of a secure transmission further and to set a foundation for a reasonable parameter choice. For the case the additional transmission will turn out to be less reliable, alternatives have been discussed. Several attacker models have been presented and corresponding countermeasures have been discussed.

In the end, a proposal for a security model is presented, that will allow the secure and reliable transmission of ADS-B messages in the known, untrustworthy environment. By combining this active security model with passive solutions presented by a number of researchers, most possible attacks on the existing environment can be prevented to allow a safe and reliable usage of aircraft in the future when using ADS-B as the main air traffic control system.

8.2. Future Work

In this thesis, a proposal for a security model to bring reliable methods for ensuring the integrity of the message and authentication of the sender was designed and discussed. But a lot of parameters this work is based on are still unknown. The most important topic is the successful injection of additional bits into a usual ADS-B message to transmit additional information without breaking the legacy reception of those messages. This project is already a work in progress within the Distributed Computer Systems group at the Technische Universität Kaiserslautern. The results of this work will determine, how the security model is to be scaled to provide a reliable and secure transmission of the information.

Further, additional parameters have to be researched. With the help and influence of pilots, manufacturers and air traffic controllers, a suitable interval duration for the key publication needs to be determined. The possibilities and requirements regarding the public key infrastructure need to be analyzed further. The suitability of the internet as a third communication channel needs to be researched regarding equipment in the aircraft, timing and security.

In the end, a real world implementation of the security model has to be build. Using software defined radios, the feasibility and reliability of the system shall be tested. When all parameters are set and a reliable way to inject additional information is found, the security model will need to be rolled out to the real world.

Bibliography

[1] I. Almomani and M. Saadeh, "Security model for tree-based routing in wireless sensor networks: Structure and evaluation." *KSII Transactions on Internet & Information Systems*, vol. 6, no. 4, 2012.

[2] S. Amin, T. Clark, R. Offutt, and K. Serenko, "Design of a cyber security framework for ADS-B based surveillance systems," in *Systems and Information Engineering Design Symposium (SIEDS), 2014*. IEEE, 2014, pp. 304–309.

[3] J. Baek, Y.-J. Byon, E. Hableel, and M. Al-Qutayri, "An authentication framework for automatic dependent surveillance-broadcast based on online/offline identity-based signature," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*. IEEE, 2013, pp. 358–363.

[4] R. Barhydt and A. W. Warren, "Development of intent information changes to revised minimum aviation system performance standards for automatic dependent surveillance broadcast (rtca/do-242a)," NASA, Tech. Rep., 2002.

[5] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management part 1: General (revision 4)," *NIST special publication*, vol. 800, no. 57, pp. 1–147, 2016.

[6] W. Bellamy III, "Airplane internet service providers leap to new generations," https://www.aviationtoday.com/2017/10/31/airplane-internet-service-providers-leap-new-generations/, accessed: 27.11.2018.

[7] P. Berthier, J. M. Fernandez, and J.-M. Robert, "Sat: Security in the air using tesla," in *Digital Avionics Systems Conference (DASC), 2017 IEEE/AIAA 36th*. IEEE, 2017, pp. 1–10.

[8] W. Blythe, H. Anderson, and N. King, "ADS-B implementation and operations guidance document," *International Civil Aviation Organization. Asia and Pacific Ocean*, 2011.

[9] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2001, pp. 437–452.

[10] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 56–67, 1998.

[11] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 1999, pp. 708–716.

[12] G. Caparra, S. Sturaro, N. Laurenti, and C. Wullems, "Evaluating the security of one-way key chains in tesla-based gnss navigation message authentication schemes," in *Localization and GNSS (ICL-GNSS), 2016 International Conference on*. IEEE, 2016, pp. 1–6.

[13] E. Chang, R. Hu, D. Lai, R. Li, Q. Scott, and T. Tyan, "The story of mode s," http://web.mit.edu/6.933/www/Fall2000/mode-s/index.html, accessed: 18.11.2018.

[14] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal," in *International Conference on Financial Cryptography*. Springer, 2002, pp. 102–119.

[15] A. Costin and A. Francillon, "Ghost in the air (traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices," *Black Hat USA*, pp. 1–12, 2012.

[16] J. Deepakumara, H. M. Heys, and R. Venkatesan, "Performance comparison of message authentication code (mac) algorithms for internet protocol security (ipsec)," in *Proc. Newfoundland Electrical and Computer Engineering Conf*, 2003.

[17] Y. Desmedt, Y. Frankel, and M. Yung, "Multi-receiver/multi-sender network security: efficient authenticated multicast/feedback," in *INFOCOM'92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*. IEEE, 1992, pp. 2045–2054.

[18] M. J. Dworkin, "Recommendation for block cipher modes of operation: The cmac mode for authentication," National Institute of Standards and Technology, Tech. Rep., 2016.

[19] European Aviation Safety Agency - Rulemaking Directorate, "Approval requirements for air-ground data link and ADS-B in support of interoperability requirements," European Aviation Safety Agency, Tech. Rep., 2012.

[20] FAA Office of Public Affairs, "Faa announces automatic dependent surveillance-broadcast architecture (press release, apa 27-02)," https://web.archive.org/web/20121022201516/http://www.faa.gov/news/press_releases/news_story.cfm?newsId=5520&print=go, 2012, accessed: 18.11.2018.

[21] C. Finke, J. Butts, and R. Mills, "ADS-B encryption: confidentiality in the friendly skies," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*. ACM, 2013, p. 9.

[22] P. Gallagher, "Digital signature standard (dss)," *Federal Information Processing Standards Publications, volume FIPS*, pp. 186–3, 2013.

[23] J. L. Gertz, "The atcrbs mode of dabs," MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, Tech. Rep., 1977.

[24] D. Hook, *Beginning cryptography with Java*. John Wiley & Sons, 2005.

[25] ICAO, "Icao pkd," https://www.icao.int/Security/FAL/PKD/Pages/default.aspx, accessed: 27.11.2018.

[26] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[27] T. Kacem, D. Wijesekera, and P. Costa, "Integrity and authenticity of ADS-B broadcasts," in *Aerospace Conference, 2015 IEEE*. IEEE, 2015, pp. 1–8.

[28] D. Knuth, *The Art of Computer Programming, Volume Two: Seminumerical Algorithms, Third Edition*. Addison-Wesley, 1997, section 4.5.4: Factoring into Primes, pp. 379–417.

[29] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," Tech. Rep., 1997.

[30] F. Kunzi, "ADS-B benefits to general aviation and barriers to implementation," Ph.D. dissertation, Massachusetts Institute of Technology, 2011.

[31] O. C. Kwon, Y. Go, Y. Park, and H. Song, "Mpmtp: Multipath multimedia transport protocol using systematic raptor codes over wireless networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1903–1916, 2015.

[32] X. Liu, S. Chandrasekhar, T. Wood, R. Tkach, P. Winzer, E. Burrows, and A. Chraplyvy, "M-ary pulse-position modulation and frequency-shift keying with additional polarization/phase modulation for high-sensitivity optical transmission," *Optics Express*, vol. 19, no. 26, pp. B868–B881, 2011.

[33] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "Raptorq forward error correction scheme for object delivery," Tech. Rep., 2011.

[34] I. A. Mantilla-Gaviria, M. Leonardi, G. Galati, and J. V. Balbastre-Tejedor, "Localization algorithms for multilateration (mlat) systems in airport surface surveillance," *Signal, Image and Video Processing*, vol. 9, no. 7, pp. 1549–1558, 2015.

[35] D. A. McGrew and S. R. Fluhrer, "Multiple forgery attacks against message authentication codes." *IACR Cryptology ePrint Archive*, vol. 2005, p. 161, 2005.

[36] P. Misra and P. Enge, "Global positioning system: signals, measurements and performance second edition," *Massachusetts: Ganga-Jamuna Press*, 2006.

[37] C. J. Mitchell, G. T. F. De Abreu, and R. Kohno, "Combined pulse shape and pulse position modulation for high data rate transmissions in ultra-wideband communications," *International Journal of Wireless Information Networks*, vol. 10, no. 4, pp. 167–178, 2003.

[38] A. Nguyen, A. Amrhar, J. Zambrano, G. Brown, O. Yeste-Ojeda, and R. Landry Jr, "Application of psk modulation in ADS-B 1090 extended squitter authentication," *International Journal of Aerospace and Mechanical Engineering*, vol. 12, no. 3, pp. 238–249, 2018.

[39] B. Nuseibeh, C. B. Haley, and C. Foster, "Securing the skies: In requirements we trust," *Computer*, vol. 42, no. 9, 2009.

[40] W.-J. Pan, Z.-L. Feng, and Y. Wang, "ADS-B data authentication based on ecc and x. 509 certificate," *Journal of Electronic Science and Technology*, vol. 10, no. 1, pp. 51–55, 2012.

[41] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," *Rsa Cryptobytes*, vol. 5, 2005.

[42] S. Pleninger and M. Strouhal, "Activities for 1030/1090 mhz spectrum saturation alleviation," *MAD-Magazine of Aviation Development*, vol. 1, no. 2, pp. 7–10, 2013.

[43] H. Rahbari and M. Krunz, "Exploiting frame preamble waveforms to support new physical-layer functions in OFDM-based 802.11 systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3775–3786, 2017.

[44] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[45] Bundesamt für Sicherheit in der Informationstechnik, "Kryptographische verfahren: Empfehlungen und schlüssellängen," https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html, BSI TR-02102-1, Version 2018-02, Tech. Rep., 2018.

[46] Committee on National Security Systems, "Cnss policy no. 15, fact sheet no. 1," NIST, Tech. Rep., 2003.

[47] Flightradar24 AB, "About - flightradar24.com - live flight tracker!" https://www.flightradar24.com/about, accessed: 18.11.2018.

[48] ICAO, "Secondary surveillance radar mode s advisory circular," Tech. Rep., 1983.

[49] International Civil Aviation Organization, "Aeronautical telecommunications, annex 10," *Catalogue of ICAO Publications*, vol. 4, 2007.

[50] Legion of Bouncy Castle Inc., "The legion of bouncycastle," https://www.bouncycastle.org/, accessed: 27.11.2018.

[51] The Apache Software Foundation, "Welcome to apache avro!" https://avro.apache.org/, accessed: 27.11.2018.

[52] The OpenSky Network, "Github - openskynetwork/osky-sample," https://github.com/openskynetwork/osky-sample, accessed: 27.11.2018.

[53] The Opensky Network, "Opensky network - facts," https://opensky-network.org/network/facts, accessed: 18.11.2018.

[54] W. R. Richards, K. O'Brien, and D. C. Miller, "New air traffic surveillance technology," *Boeing aero quarterly*, vol. 2, 2010.

[55] K. Sampigethaya and R. Poovendran, "Security and privacy of future aircraft wireless communications with offboard systems," in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*. IEEE, 2011, pp. 1–6.

[56] M. Schäfer, V. Lenders, and I. Martinovic, "Experimental analysis of attacks on next generation air traffic communication," in *Applied Cryptography and Network Security*. Springer, 2013, pp. 253–271.

[57] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up opensky: A large-scale ADS-B sensor network for research," in *Proceedings of the 13th international symposium on Information processing in sensor networks*. IEEE Press, 2014, pp. 83–94.

[58] A. Schofield, "Nav canada aims to cover major routes with ADS-B," *Aviation Week & Space Technology*, no. 07/07/2008, 2008.

[59] M. Schäfer, "Opensky's data, algorithms, and tools," https://github.com/openskynetwork/osky-sample/blob/master/oskyws15_data_tools.pdf, accessed: 27.11.2018.

[60] M. Schäfer, V. Lenders, and J. Schmitt, "Secure track verification," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 199–213.

[61] R. Shipley, "Secondary surveillance radar in atc systems: A description of the advantages and implications to the controller of the introduction of ssr facilities," *Aircraft Engineering and Aerospace Technology*, vol. 43, no. 1, pp. 20–21, 1971.

[62] A. Shokrollahi, "Raptor codes," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2551–2567, 2006.

[63] J. Song, R. Poovendran, J. Lee, and T. Iwata, "The aes-cmac algorithm," Tech. Rep., 2006.

[64] M. Strohmeier, V. Lenders, and I. Martinovic, "Lightweight location verification in air traffic surveillance networks," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM, 2015, pp. 49–60.

[65] M. Strohmeier and I. Martinovic, "On passive data link layer fingerprinting of aircraft transponders," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. ACM, 2015, pp. 1–9.

[66] M. Strohmeier, M. Schäfer, V. Lenders, and I. Martinovic, "Realities and challenges of nextgen air traffic management: the case of ADS-B," *Communications Magazine, IEEE*, vol. 52, no. 5, pp. 111–118, 2014.

[67] TheTransportTrust, "Heritage locations - south east - surrey - croydon airport," http://www.transportheritage.com/find-heritage-locations.html?sobi2Task=sobi2Details&catid=91&sobi2Id=273, accessed: 27.11.2018.

[68] travelbook.com, "Sydney-london in 20 stunden: Qatas plant neuen rekord-flug," https://www.travelbook.de/fliegen/die-10-laengsten-nonstop-flugverbindungen-der-welt, accessed:27.11.2018.

[69] P. Vabre, "Air traffic services surveillance systems, including an explanation of primary and secondary radar," http://www.airwaysmuseum.com/Surveillance.htm, accessed: 27.11.2018.

[70] S. William, "Cryptography and network security: principles and practice," *Prentice-Hall, Inc*, pp. 23–50, 1999.

[71] K. T. Wong, "Narrowband ppm semi-'blind'spatial-rake receiver & co-channel interference suppression," *European transactions on telecommunications*, vol. 18, no. 2, pp. 193–197, 2007.

[72] O. Yeste-Ojeda and R. Landry, "ADS-B authentication compliant with mode-s extended squitter using psk modulation," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE, 2015, pp. 1773–1778.

A. Appendix

A.1. Source Code CMAC generation

A.1.1. Initialization

```
public static void main(String[] args) throws Exception {
if (args.length < 2) {
        System.err.println("Usage: OskySampleReader <input file>
        ↪  <#records to read>");
        System.exit(1);
}
String filename = args[0];
int numRead = Integer.parseInt(args[1]);

// open AVRO file with pre-compiled Schema class
File f = new File(filename);
DatumReader<ModeSEncodedMessage> datumReader =
new SpecificDatumReader<>(ModeSEncodedMessage.class);
DataFileReader<ModeSEncodedMessage> fileReader =
new DataFileReader<>(f, datumReader);

ExampleDecoder decoder = new ExampleDecoder();

// allocate record for reuse to prevent
// garbage collection overhead
ModeSEncodedMessage record = new ModeSEncodedMessage();

int minTime = Integer.MAX_VALUE;
int maxTime = 0;
long msgCount = 0;
Set<Integer> serials = new HashSet<Integer>();

// Create a reference callsign and associated key
// to test against reproducible
String referenceIcao24 = "461f6c";
byte[] referenceKey =
Hex.decode("2b7e151628aed2a6abf7158809cf4f3c");
//List<byte[]> referencekeychain = new ArrayList<>();
```

```
//referencekeychain = createKeychain(10);

// generating a new private-public-keypair for the reference Icao
ECCurve curve = new ECCurve.Fp(
new BigInteger("883423532389192164791648750360308
        885314476597252960362792450860609699839"), // q
new BigInteger("7fffffffffffffffffffffff7fffffffff
        8000000000007ffffffffffc", 16), // a
new
↪  BigInteger("6b016c3bdcf18941d0d654921475ca71a9db2fb27d1d37796185c2942c0a",
↪  16)); // b
ECParameterSpec ecSpec = new ECParameterSpec(
curve,
curve.decodePoint(Hex.decode("020ffa963cdca8816ccc33b8642bedf905c3d358573
        d3f27fbbd3b3cb9aaaf")), // G
new
↪  BigInteger("883423532389192164791648750360308884807550341691627752275
        345424702807307")); // n
KeyPairGenerator g = KeyPairGenerator.getInstance("ECDSA", "BC");
g.initialize(ecSpec, new SecureRandom());
KeyPair pair = g.generateKeyPair();

// create a new hashmap storing ICAO call signs
// and assigned symmetric keys
HashMap<String, byte[]> keys = new HashMap<>();
HashMap<String, Integer> counter = new HashMap<>();

// initialize SecureRandom once, since it's
// expensive to initialize
SecureRandom random = new SecureRandom();

//Default key for testing purposes
byte[] keyBytes128 =
Hex.decode("2b7e151628aed2a6abf7158809cf4f3c");

//create a new csv file to store the hashes in
PrintWriter pw = new PrintWriter(new File("hashes.csv"));
StringBuilder sb = new StringBuilder();
sb.append("icao24");
sb.append(",");
sb.append("key");
sb.append(",");
sb.append("msg_hex");
sb.append(",");
```

```
sb.append("msg_bin");
sb.append(",");
sb.append("mac_hex");
sb.append(",");
sb.append("mac_bin");
sb.append('\n');
```

A.1.2. CMAC creation

```
while (fileReader.hasNext()) {
// get next record from file
record = fileReader.next(record);

msgCount++;
serials.add(record.getSensorSerialNumber());
maxTime = Math.max(record.getTimeAtServer().intValue(), maxTime);
minTime = Math.min(record.getTimeAtServer().intValue(), minTime);

if (--numRead >= 0) {
        ModeSReply msg;
        try {
                msg =
                ↪  Decoder.genericDecoder(record.getRawMessage().toString());
        } catch (BadFormatException e) {
                System.out.println("Malformed message! Skipping it.
                ↪  Message: " + e.getMessage());
                continue;
        } catch (UnspecifiedFormatError e) {
                System.out.println("Unspecified message! Skipping
                ↪  it...");
                continue;
        }
        // For the example implementation, we are only interested in
        ↪  position reports
        //        case ADSB_AIRBORN_POSITION:
        if(msg.getType() == ModeSReply.subtype.ADSB_AIRBORN_POSITION){
                //print the information included
                System.out.println("record #" + msgCount + " (raw
                ↪  AVRO)");
                for (Field field : record.getSchema().getFields()) {
                        if (record.get(field.name()) != null)
                                System.out.printf("\t%-20s %s\n",
                                ↪  field.name() + ":",
                                record.get(field.name()));
```

```
        }
System.out.println("record #" + msgCount + "
↪  (decoded)");

byte[] Icao24 = msg.getIcao24();
String icao24 = tools.toHexString(Icao24);

//initiate the mac-engine
BlockCipher cipher = new AESFastEngine();
Mac mac = new CMac(cipher, 128);

//find the key for this plane or create a new one if
↪  unknown
if (icao24.equals(referenceIcao24)){
        keyBytes128 = referenceKey;
        System.out.println("Icao24 " + referenceIcao24
        ↪  + ", using reference Key: " +
        ↪  tools.toHexString(referenceKey));
        if(!counter.containsKey(icao24)){
                counter.put(icao24, 1);
        } else{
                int count = counter.get(icao24);
                counter.put(icao24, count+1);
        }
} else if (!keys.containsKey(icao24)) {
        byte[] keyBytes = new byte[16];
        random.nextBytes(keyBytes);
        keyBytes128 = keyBytes;
        keys.put(icao24, keyBytes);
        counter.put(icao24, 1);
        System.out.println("Key for " +
        ↪  tools.toHexString(Icao24) + " not found. "
        ↪  + keys.size() + " Elements in map");
} else {
        keyBytes128 = keys.get(icao24);
        int count = counter.get(icao24);
        counter.put(icao24, count+1);
        System.out.println("Key for " +
        ↪  tools.toHexString(Icao24) + " found: " +
        ↪  tools.toHexString(keyBytes128));
}

KeyParameter key = new KeyParameter(keyBytes128);
```

```
mac.init(key);

//////// decode the message, add information to CSV,
↪  print summary
String raw = record.getRawMessage().toString();
System.out.println("Flight number " + icao24 + ":");

sb.append(icao24);
sb.append(",");

String currentKey = tools.toHexString(keyBytes128);
sb.append(currentKey);
sb.append(",");

System.out.println("raw: " + raw);

sb.append(raw);
sb.append(",");

String raw_bin =
↪  hexToBin(record.getRawMessage().toString());
System.out.println("raw_bin: " + raw_bin);

sb.append(raw_bin);
sb.append(",");

//*************************************
// Calculate the CMAC
byte[] toMacMsg = Hex.decode(raw);
mac.update(toMacMsg, 0, toMacMsg.length);
byte[] out = new byte[16];
mac.doFinal(out, 0);
System.out.println("Mac: " + hexToBin(new
↪  String(Hex.encode(out))));
System.out.println("Mac_Hex: " + new
↪  String(Hex.encode(out)));
System.out.println();

sb.append(new String(Hex.encode(out)));
sb.append(",");
sb.append(hexToBin(new String(Hex.encode(out))));
sb.append('\n');

//*************************************
```

```
                decoder.decodeMsg(record.getTimeAtServer(),
                ↪  record.getRawMessage().toString(), null);

        }
} else break;

}
```

A.2. Source Code Verifier

```
public static void main(String[] args) throws IOException {
        readCSV(args[0]);
}

private static void readCSV(String pathToCSV) throws IOException
{
        String referenceIcao24 = "461f6c";
        byte[] referenceKey =
        ↪  Hex.decode("2b7e151628aed2a6abf7158809cf4f3c");
        int false5 = 0, false10 = 0, false15 = 0, false20 = 0;
        String mac_calced;

        String path = pathToCSV;
        try{
        Reader reader = Files.newBufferedReader(Paths.get(path));
        CSVReader csvReader = new CSVReader(reader);
        String[] nextRecord;


        PrintWriter pw = new PrintWriter(new File("verifier.csv"));
        StringBuilder sb = new StringBuilder();
        sb.append("icao24");
        sb.append(",");
        sb.append("msg_hex");
        sb.append(",");
        sb.append("mac_msg");
        sb.append(",");
        sb.append("mac_calc'd");
        sb.append(",");
        sb.append("pos");
        sb.append(",");
        sb.append("f/pos 5");
```

```
sb.append(",");
sb.append("f/pos 10");
sb.append(",");
sb.append("f/pos 15");
sb.append(",");
sb.append("f/pos 20");
sb.append(",");
sb.append("#equals");
sb.append('\n');

while ((nextRecord = csvReader.readNext()) != null){
        //skip the titles
        if (nextRecord[0].equals("icao24")) continue;

                sb.append(nextRecord[0]);
                sb.append(",");
                sb.append(nextRecord[2]);
                sb.append(",");
                sb.append(nextRecord[4]);
                sb.append(",");

                mac_calced = new
                ↪  String(Hex.encode(calculateMAC(nextRecord[2],
                ↪  referenceKey)));
                sb.append(mac_calced);
                sb.append(",");
                int comparetrunk = 128;

                if(nextRecord[0].equals(referenceIcao24) &&
                ↪  nextRecord[4].equals(mac_calced)){
                sb.append("yes,no,no,no,no");
        } else if(nextRecord[0].equals(referenceIcao24) &&
        ↪  !nextRecord[4].equals(mac_calced)){
                System.out.println("Something's wrong,
                ↪  aborting...");
                System.exit(0);
        } else{
                sb.append("no");
                sb.append(",");
                comparetrunk =
                ↪  compareTrunk(nextRecord[4],mac_calced);
                if(comparetrunk >= 5){
                        sb.append("yes,");
                        false5 ++;
```

```
                        } else sb.append("no,");
                                if(comparetrunk >= 10){
                                sb.append("yes,");
                                false10++;
                        } else sb.append("no,");
                                if(comparetrunk >=15){
                                sb.append("yes,");
                                false15++;
                        } else sb.append("no,");
                                if(comparetrunk >= 20){
                                sb.append("yes");
                                false20++;
                        } else sb.append("no");
                }

                sb.append(",");
                sb.append(comparetrunk);
                sb.append('\n');
        }
        sb.append('\n');
        sb.append("sum,,,,," + false5 + "," + false10 + ";" + false15 +
        ↪  "," + false20 + ",");

        csvReader.close();
        reader.close();
        pw.write(sb.toString());
        pw.close();

        } catch (IOException e) {
                e.printStackTrace();
        }

}

//compares the single bits of two macs and returns for how many bite the
↪  macs are equal
private static int compareTrunk(String mac, String verifier){
        mac = hexToBin(mac);
        verifier = hexToBin(verifier);
        int counter = 0;
        while(mac.charAt(counter) == verifier.charAt(counter)) {
                counter++;
        }
return counter;
```

```
}

private static byte[] calculateMAC(String message, byte[] keyBytes){
        BlockCipher cipher = new AESFastEngine();
        Mac mac = new CMac(cipher, 128);
        KeyParameter key = new KeyParameter(keyBytes);
        mac.init(key);
        byte[] toMacMsg = Hex.decode(message);
        mac.update(toMacMsg, 0, toMacMsg.length);
        byte[] out = new byte[16];
        mac.doFinal(out, 0);

        return out;
}

private static String hexToBin(String s) {
        String preBin = new BigInteger(s, 16).toString(2);
        //padd with leading zeros
        if((preBin.length()%8) != 0){
                String zeros = "00000000";
                int topad = (preBin.length()%8);
                preBin = zeros.substring(topad) + preBin;
        }
        return preBin;
}
```