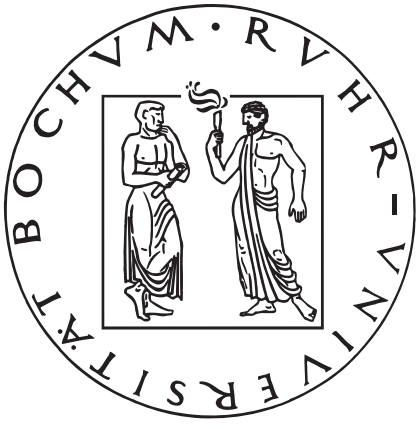
Security mechanisms for positioning systems - enhancing the security of eLoran

Georg T. Becker

July 30, 2009

Master Thesis
Ruhr-Universität Bochum



Chair for Embedded Security
Prof. Dr.-Ing. Christof Paar

Written at the GPS Laboratory at Stanford University
Co-Advised by Dr. Sherman Lo (Stanford University)

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Bochum, den July 30, 2009 _____

Abstract

The Positioning, Navigation and Timing (PNT) infrastructure becomes more and more important. Today, most PNT systems are based on the Global Positioning System (GPS). This dependency on GPS makes the PNT infrastructure very vulnerable. Therefore, a backup for GPS is needed in case of attacks on GPS or system failures. One of the most promising backups for GPS is the Long Range Navigation System (LORAN). To be able to be an independent backup for GPS, LORAN is currently being upgraded to enhanced LORAN (eLoran). Civil navigation systems are very vulnerable to jamming and spoofing attacks, as no security countermeasure is used to prevent these attacks. But because these systems are used in more and more critical applications, the threat of attacks and therefore the need for security mechanisms increases. The development of eLoran is a great opportunity to embed security mechanisms into the design of eLoran to make eLoran a secure positioning system.

In this thesis, efficient security mechanisms for eLoran are discussed. A modified version of the TESLA authentication algorithm, called adjusted TESLA, is proposed for the eLoran data channel. The main modification is to embed the transmission time of each key into the one-way chain generation. With this modification it can be shown that a key size of 80 bit can provide sufficient security for many years. Furthermore, it is possible to use this key on its own to authenticate the most important information in eLoran, the source of the signal and the transmission time of the signal. In this way, it is not necessary to use a MAC to prevent signal-synthesis attacks.

But message authentication is only one step towards a secure positioning system. Only signal-synthesis and counterfeit correction message attacks can be prevented using adjusted TESLA. Other attacks like selective-delay and relaying attacks are still possible. Therefore, additional countermeasures against these attacks are discussed in the last chapter. A new method, called colliding signals, is introduced that can prevent signal-synthesis attacks. Although this method seems to be impracticable for eLoran, it can be a powerful tool in other terrestrial positioning systems.

Contents

List of abbreviations

AES	Advanced Encryption Standard
DES	Data Encryption Standard
DGPS	Differential Global Positioning System
DLP	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
DSSS	Direct Sequence Spread Spectrum
ECDSA	Elliptic Curve Digital Signature Algorithm
eLoran	enhanced LORAN
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GRI	Group Repetition Interval
IMU	Inertial Measurement Unit
LAAS	Local Area Augmentation System
LBS	Location Based Service
LDC	LORAN Data Channel
LORAN	LOng RAnge NAvigation system
MAC	Message Authentication Code
PNT	Positioning Navigation and Timing system
RF	Radio Frequency
RS	Reed-Solomon coding
TDOA	Time Difference Of Arrival
TESLA	Timed Efficient Stream Loss-tolerant Authentication)
TOA	Time Of Arrival
WAAS	Wide Area Augmentation System

1 Introduction

1.1 Motivation

In todays world, many different applications depend on aspects of positioning, navigation and timing (PNT) systems. These applications range from car-navigation systems, to location based services (LBS) that show the closest restaurants on your mobile phone, to guiding aircrafts in precision approach landing operations. Not only does the accuracy and availability of PNT systems such as the Global Positioning System (GPS) grow, but receivers are also becoming cheaper and smaller. Today, mobile phones already have embedded GPS receivers. All this will result in even more location based services in all kinds of different areas. However, with the increasing use of PNT services, the threat of misusage increases as well. The PNT infrastructure is already considered as one of the most critical infrastructures, as many different infrastructures, from the transportation infrastructure over the IT infrastructure to the power supply rely on PNT services. [4]
The big breakthrough of PNT systems was reached with the US GPS system. Since GPS became operational, its accuracy and availability is continuously growing. Other Global Navigation Satellite Systems (GNSS) such as European's GALILEO system, Russia's GLONASS system or China's COMPASS system will introduce redundancy and thereby also increase the accuracy and availability of GNSS systems. Much research was done to improve GNSS systems for life critical applications such as precision approach operations for aircraft landings. Augmentation systems such as LAAS or WAAS have been designed to improve GNSS systems by providing correction and warning messages. However, although GNSS systems are embedded in more and more critical applications, the security against attackers has not gained a lot of attention in the public yet. Attacking a GNSS system such as GPS is quite easy, as no countermeasures for attacks against the civil navigation channel are implemented yet. GNSS signals are very weak, so that jamming or spoofing these signals can be done with cheap of-the-shelf hardware. As a matter of fact, GPS jammers with different power outputs can already be purchased easily over the Internet. As a recent analysis showed [10], most current GPS receivers do not even support the most rudimentary spoofing countermeasures.
Because of GPS's crucial role in todays infrastructure, in 2001 the Volpe report emphasizes the need for a backup for GPS. [4] According to many scientists, the most promising backup for GPS is the LOng RAnge Navigation system (LORAN). LORAN is a terrestrial, low-frequency, high power navigation system built in the 50s by the US army. It covers most parts of the western hemisphere. Due to its design, it has very different error sources compared to GPS, making it a perfect backup system for GPS. Because LORAN uses high power, low-frequency signals, jamming

LORAN is much more difficult than jamming GPS. LORAN has also the advantage that due to the long wavelength of the LORAN signals, the signals can penetrate houses and can therefore be used in urban territories. To meet the increased accuracy and availability requirements of todays positioning systems, recently much effort and money has been spent to develop an improved version of LORAN, called enhanced LORAN (eLoran). The development of eLoran is still an ongoing process. Many LORAN transmitters have been modernized and upgraded to support the new navigation signals. This new version of LORAN will meet all performance requirements needed to be an independent backup for GPS.
Changing navigation systems is a very time consuming task, as the infrastructure is very expensive and will only be exchanged slowly. Furthermore, all changes need to be backward compatible to support the existing equipment. All this makes it hard to embed new security mechanisms into current navigation systems such as GPS. Even if the decision to add a security mechanism to GPS is made today, it would probably take at least 20 years until all satellites support this new security feature. [30] The development of eLoran is therefore a great chance to increase the security of LORAN by adding security mechanisms such as authentication into the design of eLoran. These security mechanisms can be crucial in the future to protect the PNT infrastructure.
Furthermore, a secure positioning system can lead to many other applications that are not possible without appropriate security mechanisms. If the security of current positioning systems can be increased, it might be possible to use location as a security parameter, rather than only using security mechanisms to increase the security of positioning systems. For example, location verification could be used as an additional authentication mechanism, preventing impersonation attacks or enabling location based access control.

1.2 Organization of this thesis

This thesis discusses how the security of eLoran can be increased efficiently. In chapter 2, the different attacks on navigation systems are introduced and categorized. The third chapter consists of an overview over all known countermeasures against these attacks. In the fourth chapter, a short introduction to LORAN and the improvements that lead to eLoran is given. In chapter 5, the possible message authentication algorithms that can be used in connection with eLoran are discussed. I developed a modified version of the Timed Efficient Stream Loss-tolerant Authentication algorithm (TESLA), called adjusted TESLA. The comparison of the different message authentication methods shows that adjusted TESLA is the best choice for eLoran. The efficient implementation of this adjusted TESLA in eLoran is discussed in chapter 6. My implementation is directly designed for the needs of eLoran. By embedding the time and station ID message into the generation of the one-way chain, this message can be authenticated without the need of a MAC. In this way, the most important information, the source and transmission time of the signal, can be authenticated by verifying the one-way chain keys. This can result in an increase in authentication time or bandwidth of up to 50%. Furthermore, my security anal-

ysis of adjusted TESLA in chapter 6 shows that by adding a timestamp into the one-way generation, in positioning systems adjusted TESLA is much more secure than the original TESLA. Besides the increased complexity for breaking the keys for one one-way chain, it will only be possible for an attacker to break the chain for a limited time period, due to the timestamp. Even if attacking one chain becomes comutational feasible, it will still be impossible to break the keys of more than one one-way chain at the same time. This makes it impossible for an attacker to forge signals from more than one eLoran station. But at least three signals are needed to launch a signal-synthesis attack.
Message authentication is only the first step towards a secure positioning system. So far no security mechanism that can prevent selective-delay attacks is known for terrestrial, low-frequency positioning systems such as LORAN. In chapter 7, I will introduce a new security mechanism called colliding signals. Colliding signals can prevent the powerful selective-delay attacks by ensuring that different data is lost at different locations. To the best of my knowledge, colliding signals is the first security mechanism that uses this strategy to defend signal-synthesis attacks The thesis is completed by a conclusion in Chapter 8.

2 Attacks on positioning systems

There are several different ways how positioning systems can be attacked. In this chapter, the different types of attacks are described. The attacks do not focus on a particular positioning system and can be applied to almost all global positioning systems. This thesis focuses on passive positioning systems, but most of the described attacks are also a threat to active positioning systems.[1]

2.1 Jamming

The first attack is the most trivial, but also the most common attack on positioning systems. A jamming attack is a denial-of-service attack (DOS) against positioning systems. The attacker tries to interfere the positioning messages transmitted by the positioning system, so that the receiver can not determine a correct position. One way to achieve this, is to raise the noise level of the transmission until the navigation signals can not be detected by the receiver any more. Another way to jam a receiver is to send out misleading navigation messages, so that the receiver can not determine which navigation messages are valid and which are faked. This leads to contradictions during the position calculation and the receiver will not be able to compute a position.
Jamming attacks are a great threat to positioning systems, especially to satellite based positioning systems such as GPS or GALILEO. This is due to the fact that the received signals from the satellites are very weak. Different reports have shown the vulnerability of these positioning systems. Areas of several kilometers were spoofed with small transportable equipment[6]. Several armies possess GPS jammers. An example of the use of GPS jammers by military was given in 2003 in the Iraq war. The Iraqi military tried to use Russian GPS jammers to jam GPS guided weapons and other military GPS receivers. But GPS jammers are not limited to military access. Small GPS jammers which you can plug into the car's cigarette lighter are already advertised and sold in the Internet for less than $50. Furthermore, detailed instructions for building GPS jammers as small as a pack of cigarettes are publicly available on the Internet.
Besides intentional jamming, positioning systems can sometimes also be jammed unintentionally. A very good example of an unintentional jamming was an incident at Moss Landing Harbor in April 2001. GPS was jammed for an area of about 1 square kilometer in Moss Landing Harbor, a moderate-sized harbor about 100 kilometers south of San Francisco, for about two months. During this time, ships

[1] In an active positioning system the user is able to communicate with the transmitter stations, while in a passive positioning system the user only receives signals from the transmitter.

were not able to use GPS to navigate into the narrow harbor entrance. Locating the jamming device turned out to be very challenging. Finally, researchers identified the interference source as three active UHF/VHF TV antennas. [5] (Each of the antennas on their own caused enough interference to jam GPS within the Harbor) This incident shows how real the threat of GPS jamming is. Although the jamming was not intentional, it was very hard and time consuming to locate and remove the interference sources. The fact that it were not sophisticated spoofing devices, but only male functional commercial TV antennas that cause such big problems, shows how vulnerable GNSS systems are to jamming attacks. Detecting sophisticated spoofing devices might turn out to be even more challenging.
Jamming a terrestrial positioning system like LORAN is much harder compared to jamming a satellite based positioning system. This is due to the fact that LORAN uses a high-power, low-frequency signals. To jam LORAN, the attacker needs to overcome the strong LORAN signal. However, to efficiently transmit a low-frequency signal, high antennas are needed. This makes a jamming attack against LORAN very challenging. Detecting a LORAN jammer is quite easy, as the jammer needs to transmit high power signals that are very easy to track. A detailed analysis of the difficulties in jamming LORAN is given in section 4.4.

2.2 Signal-synthesis attack

In a signal-synthesis attack, an attacker generates and sends out false navigation signals to make the receiver believe to be at a different position. If the structure of the navigation message is known to the public, an attacker can easily create valid navigation messages. For example, an attacker can simply attach a power amplifier and an antenna to a commercial civilian GPS signal simulator, which are used for testing GPS receivers, to generate the false navigation signals. Attacks using a signal generator without any modifications might be easy to detect, as generated GPS signals are very likely unsynchronized with the original GPS signal. This could cause the receiver to loose track of the GPS signal, which can be used as a sign for an attack. This unsynchronized attack will very likely cause a jump in time and signal strength, which is a very good indicator for an attack, as well. However, more sophisticated GPS spoofing devices can already be build with of-the-shelf hardware as described in [10]. These spoofing devices synchronize the signals with the original GPS signals and slowly take control over the transmission channel. This will make it very difficult to detect the attacks, as there are neither jumps in signal-strength, phase or frequency, nor does the receiver loose track of the GPS signals.
Just like jamming, an over-the-air signal-synthesis attack against terrestrial positioning systems such as LORAN is much more difficult than an attack against a satellite-based positioning system such as GPS. This is again due to the fact that terrestrial systems use high-power, low-frequency signals, and that an attacker therefore needs much more energy and bigger antennas to take over the signal.
However, if the attacker has physical access to the receiver, the attacker could simply plug the signal simulator directly to the receivers antenna. In this case, the attacker would gain full control over the communication channel and it would make

no difference if a terrestrial or satellite based positioning system is used. In many scenarios the user can be the potential attacker. If the positioning system is used for monitoring or access control, the user might be very interested in spoofing the receiver. For example, the driver of a truck that wants to make an illegal stop or the fishing boat that wants to fish in a restricted area are potential attackers that have full physical access over the receiver.

2.3 Relaying attack (wormhole attack)

The basic idea of a relaying attack is to relay the navigation signals received at the wanted spoofing position p' to the receiver at the actual position p. This will make the receiver believe to be at the false position p', although the true position of the receiver is p. There are different ways to execute a relaying attack. If the attacker has physical access to the receiver, an attacker can dismount the antenna and connect the receiver to an antenna located at the false position p'. If the distance between the wanted spoofing location and the actual location is very big, an attacker will very likely not directly attach the antenna at the false position p' to the receiver but will use another channel to transmit the received signals at position p' to the receiver at position p. This kind of attack is also often called wormhole-attack.

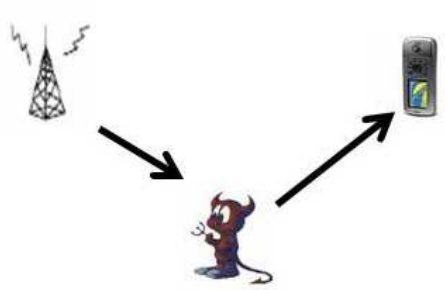


Figure 2.1: An illustration of a relaying attack. The attacker relays the signals received at his location p' to the victim's receiver at location p. The victim will falsely believe to be at the attacker's location p'.

Besides the logistical problems of receiving and transmitting the signals from position p' to position p, the attacker has to overcome the problem that due to the relaying the signals will be older than the correct signals and therefore the receiver clock might locate a jump in time. Depending on the distance and used equipment, this time difference might be big enough to warn the receiver. There are several different ways the attacker can try to eliminate this time offset so that the receiver does not get suspicious.
One was to do this is to jam the receiver until the clock offset is big enough or to use other methods to offset the clock of the receiver, for example by changing the operation temperature of the clock. Another option the attacker has to offset the receiver clock is to slowly introduce a delay to the navigation signal received at position p. The receiver always calibrates its clock according to the navigation signals.

In this way the attacker can slightly increase the delay over some time, until the clock of the receiver has the needed offset. As every clock has some offset after some time, the receiver will not get suspicious. Only big offsets during a short period of time are suspicious and warn the receiver of a relaying attack. When the receiver clock has the needed offset, the attacker can start sending the signals from the false position p'. This attack will not be detected by measuring time jumps, as there is no current jump in time. However, there is a jump in position which can also be detected. In a more sophisticated attack, the attacker might slowly increase the distance between p and p' so that there is no jump in position. But this significantly increases the complexity of the attack.
Over-the-air relaying attacks against GPS over small distances are already very easy to achieve without the need of any engineering skills. This is due to the fact that GPS repeaters, which replay the GPS signals received at one location at another location, are commercially available and can be legally (at least in some countries) bought. These repeaters do exactly what is done in a relaying attack. An example of a commercial GPS repeater is shown in figure 2.2. GPS repeaters are used to enable GPS acquisition in buildings where the original GPS signals are too weak. These GPS repeaters are sold with different cable lengths form 3 meter up to 30 meters. GPS repeaters retransmit the GPS signals over radio, so that no connection between the receiver and the GPS repeater is needed. Hence, the attacker would not need to have physical access to the receiver to launch a relaying attack. Because the GPS repeater operates in the GPS frequency band, the use of a GPS repeater is illegal in some areas or countries as the use might violate the FCC regulations. To avoid these regulations, there are also shielded GPS repeater available. These devices shield the radio signals going in and going out of the GPS repeater. These shielded GPS repeaters are especially interested for attackers having physical access over the receiver, as the original GPS signals will not be detectable by the receiver any more.

Figure 2.2: The commercial GPS repeater GPSRKL12 from GPS SOURCE.

2.4 Selective-delay attack

In most RF-based positioning systems, the position is calculated by either time of arrival (TOA) of the navigation signals, or the time difference of arrival (TDOA) of the navigation signals from different transmitters. In time of arrival, the receiver uses the time it took the navigation signal to reach the receiver to compute the distance between the transmitter and the receiver. By computing the distance between three different transmitters, the receiver can compute its three-dimensional position. To determine the travel time of the signals, the receiver needs to be synchronized with the transmitter stations. If the receiver's clock is not synchronized with the transmitter station, the receiver can use a fourth signal to determine the exact time. Hence, 4 signals are needed to determine the three-dimensional position, as in most cases the receiver will not be synchronized with the transmitter station. Time-difference-of-arrival (TDOA) works similar to TOA. The difference is that the receiver does not use the arrival time of one signal to compute the position but the difference in arrival times between two signals from two different transmitters. A more detailed description of TOA and TDOA is given in chapter 4. A selective-delay attack works with TOA as well as with TDOA

In a selective-delay attack, an attacker takes advantage of the fact that the arrival time of the navigation signals are used to determine the position. An attacker delays each navigation message in a way that the receiver calculates a false position. The most simple attack would be to delay all signals for the same amount of time, e.g. by adding a cable between the receiver and the antenna. However, if all signals are delayed for the same amount of time, this will only result in a different clock offset. The position will be the same. Hence, a simple delay attack is an efficient way to attack time synchronization but has no direct impact on positioning. It can only impact the positioning if the user is moving fast and the delay is very long. In this case, the receiver will calculate an old position that does not match the current position.

In comparison to this simple delay-attack, in a selective-delay attack each signal is not delayed for the same amount of time. Assume that the attacker is at position P_A. The victims receiver is at position P_R and the attacker wants to make the receiver believe to be at the false position P_F. Furthermore, assume that there are 4 navigation signals available. The attacker receives the signals S_1 at time t_1, S_2 at time t_2, S_3 at time t_3 and S_4 at time t_4. The attacker now calculates for each signal S_i the time t_i' at which the receiver needs to receive each signal, such that the receiver determines P_F as its current position. The attacker retransmits each signal S_i with a delay $t_i' - t_i$. However, $t_i' - t_i$ needs to be a positive number. Therefore, every message is also delayed for a fix time x such that the delay is $t_i' - t_i + x \geq 0$. This will result in an clock offset at the receiver of the amount of x. This offset might be detectable by the receiver. But the attacker can use the same techniques as described in the relaying attack to circumvent this problem. He can either tamper the receiver clock so that the receiver accepts this offset, or slowly introduce a delay into the signal until the needed offset is reached. Figure 2.3 illustrates a selective-delay attack.

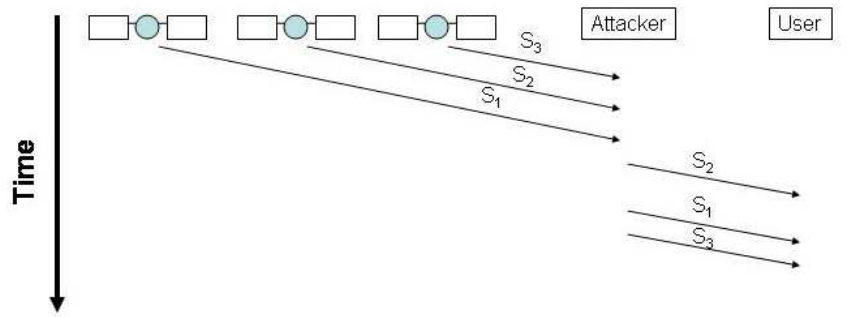


Figure 2.3: Illustration of a selective-delay attack. The signals from the satellites are delayed for different amounts of time so that the user will compute a false position.

2.5 Counterfeit correction message attack

Positioning systems can be effected by several different error sources. For example, these errors can be clock errors in the transmitter stations, multipath effects or atmospheric effects. To minimize the effect of these error sources, reference stations are used to collect data and to generate correction messages that will help the receiver to calculate a more accurate position. Famous examples for such augmentation systems for GPS are the differential GPS (DGPS), the Wide Area Augmentation System (WAAS), or the Local Area Augmentation System (LAAS). Enhanced LORAN (eLoran) will also support correction messages to be able to achieve the required accuracy and availability. The correction messages are either transmitted over a different channel, like it is done in DGPS, WAAS or LAAS, or are part of the data message transmitted by the positioning system, like it is planed for eLoran. These correction messages can typically have impacts of at most 200-600 meters. But most of the time the correction messages will only change the calculated position for a few meters.
In a counterfeit correction message attack, the attacker forges these correction messages. A receiver using these faked correction messages will compute a false position. Besides directly forging the correction message, the attacker can also try to attack the reference stations. If the attacker jams the reference stations, no accurate correction messages can be generated, so that the receiver will not be able to generate a position as accurate as it is possible with correction messages. In the case of eLoran, these correction messages can be essential to calculate an accurate position. More problematic are spoofing attacks on reference stations. If the attacker is able to attack a reference station using a signal-synthesis attack, a selective-delay attack or some other kind of attack, the reference station will generate false correction messages. If these false correction messages are not detected by other reference stations, these false correction messages will be sent out, causing the receivers to compute false positions.

2.6 Shifting the tracking point

Another way of attacking a positioning system is described in [18]. The important information in positioning systems is the exact arrival time of a signal. The arrival time of a signal is defined by a tracking point. In the case of LORAN, this tracking point is defined as the sixth zero crossing of a LORAN pulse. The attacker can try to overlay a signal on the original LORAN pulse, so that the receiver will falsely detect a wrong tracking point. In this way, the attacker does not need to overcome the signal power so that much less power is needed for the attack. Whether or not such an attack is possible strongly depends on the signal design of the positioning system. Overlaying a signal over the original signal might result in a false envelope shape of the corresponding signal and therefore might be detectable. A more detailed description of this attack on eLoran can be found in section 4.4.

2.7 Tamper the receiver

If the attacker has access to the receiver, he can try to tamper the receiver. There are various different ways how the attacker can try to tamper the receiver. This depends on the attack scenario. One possible attack is to change the firmware of the receiver, so that false positions are calculated. If the receiver uses a display as an output, the attacker might tamper the receiver by simply exchanging the receiver hardware in the insight of the receiver with a device that creates faked outputs and might be controlled via a radio link by the attacker. Other attacks can include tampering the receiver clock to enable other attacks such as relaying attacks. As these attacks depend on the used hardware and the environment in which this hardware is used, these types of attacks are not further discussed in this paper.
However, it should be noted that tampering attacks on receivers are especially dangerous if the receiver and the transmitter stations share a common secret. For example, the security of the GPS P(Y) code used by the US military is based upon a secret key. This secret key is embedded in every military GPS receiver. If an attacker would be able to successfully tamper a military GPS receiver and reveal this key, the entire military GPS P(Y) code would become insecure. Hence, tampering one device would have influence on all other used devices, even if these devices are tamper proof.

3 Security mechanisms for positioning systems

There are several countermeasures against the different attacks. These countermeasures differ in complexity and in the amount of provided security. Some of the countermeasures can be implemented on the receiver side, others need modifications of the transmitted signals. In the following, the basic concepts of the different countermeasures are introduced. These concepts can be applied to most positioning systems as they describe only the basic idea, and not the implementation.

3.1 Signal and data observation

Signals sent out by the attacker, instead of the valid transmitters, very likely differ in several properties. By monitoring and comparing these properties, the receiver might be able to detect the forged signals. Whether a receiver is able to detect an attack or not depends on how sophisticated the attack is. The following analysis helps to detect an attack:[10]

- Amplitude discrimination
  Jumps in amplitude and signal-to-noise ratio of the navigation signal can be used to identify possible attacks. If unusual strong navigation signals arrive, the receiver can reject these signals as they might be spoofed.

- Time-of-arrival discrimination
  Unsynchronized attacks will create a jump in arrival times of the navigation signals and therefore create a clock-offset. A big clock offset during a short time is a good indicator for an attack. Furthermore, if the phase of the signals change quickly, this can also be seen as an indication for an attack.

- Consistency of navigation inertial measurement unit (IMU) cross-check
  Inertial measurement units (IMU) track motions using a combination of accelerometers and gyroscopes. Knowing the start location, this data can be used to compute the current position. Hence, it is a redundant source of positioning. By comparing the data of the IMU with the GPS location, spoofing attacks on GPS can be detected. Obviously, this countermeasure significantly increases the receiver's complexity and cost.

- Polarization discrimination
  Receivers can check if the navigation signals have the correct polarization. Unsophisticated attackers might send out signals with a different polarization than the original navigation signals.

- Angle-of-arrival discrimination
  As the navigation signals are transmitted by different transmitters, the angle-of-arrival of the different signals differ from each other. However, in most cases an attacker will only transmit from one location. Hence, the angle of arrival of these spoof signals all come from the same direction. Using array antennas, the receiver can check if the signals come from the expected directions.

- Vestigial signal defense
  The attacker will most likely not be able to suppress the authentic navigation signal if he does not have physical access to the receiver. In the case of GNSS systems, the attacker would need to have centimeter-level knowledge of the 3-dimensional vector between the target antenna and the attacker's transmitter to transmit an effective suppressor signal. [10] Hence, a powerful countermeasure against spoofing is to check for the remainder of the authentic navigation signal.

- Jumps in space
  A very simple countermeasure is to check for unusual jumps in positions. Jumps of several kilometers within milliseconds are obviously wrong. In signal-synthesis attacks the attacker can easily circumvent this countermeasure by slowly introducing a position error. However, checking for jumps in space can be very efficient against relaying attacks, as in this case it might not be possible for the attacker to avoid abnormal jumps of the position.

These signal analyses help to defend against signal-synthesis, selective-delay and relaying attacks. However, the more sophisticated an attack is, the more likely will these countermeasures fail. The big advantage of these methods is that they can be implemented on the receiver side. Hence, these countermeasures do not require to change the positioning system. The effort needed to implement the countermeasures differs a lot. For example, the IMU cross check needs expensive hardware and complex calculation and is only reasonable in high-security applications. On the other hand, checking for jumps in time and space can easily be done without additional hardware. It is up to the manufacturer to decide how much security is needed and which methods he wants to implement into his receivers. According to the application, each receiver might consists of different security levels. Unfortunately, many manufacturers have not realized the threat of attacks on positioning systems yet. Tests and interviews showed that most of the current receivers on the market do not include even the most rudimentary countermeasures. [10]

The most promising signal and data observation technique is the angle of arrival check. If an arrayed antenna is used, the receiver will be able to determine the angle of arrival of the incoming signals. If all signals are transmitted from the same transmitter, the receiver will reject these signals. Therefore, the attacker needs several spoofing devices located in such a way that the receiver accepts the angle-of-arrival of these signals. Besides the logistical difficulty of setting up several spoofing devices, the attacker also needs to synchronize the transmission of each spoofing device.

Hence, angel-of-arrival discrimination is one of the most promising countermeasures against spoofing attacks.

3.2 Message Authentication

The goal of message authentication is to prevent an attacker from generating his own navigation messages. This will make signal-synthesis attacks and counterfeit-correction message attacks impossible. In most cases, message authentication is a requirement for further countermeasures such as hidden markers. Message authentication can be achieved in three different ways, encryption with an integrity check, message authentication codes (MAC) and digital signatures.

3.2.1 Encryption with an integrity check

If the navigation signal consists of digital data, this data can be encrypted. To protect against signal-synthesis attacks the user needs to be sure that the data messages were generated and send by the transmitter station. For additional protection against counterfeit correction message attacks, it should not be possible to change any part of the data messages. This can be done using encryption with an integrity check. The integrity check can consist of a hash of the data message. The data message with the integrity check is encrypted with a symmetric cipher. If an attacker changes parts of this ciphertext or tries to create his own ciphertext without the knowledge of the correct key, the receiver can detect this fraud because the decrypted message will not pass the integrity check.
It is also important to prove the freshness of the cipher texts. Otherwise an attacker can use old encrypted navigation messages. As these messages were encrypted by the transmitter, they will pass the integrity check. Such old navigation messages will very likely lead to false positions. One way to protect against such replay attacks is to add a timestamp to the data before encrypting it. In this way, a receiver will be able to distinguish new and old data messages.
In a symmetric encryption scheme, the same key is used for encryption and decryption. Every receiver needs the key to decrypt the cipher. However, this means that the key in each receiver can be used to not only decrypt data, but also to encrypt data. If an attacker gains access to the key, he can generate valid navigation messages. Hence, the security of this system relies on keeping the key secret. With a growing size of users this becomes a very difficult task. Therefore, encryption is only an option for a limited and trusted user group such as the military.
If the data is encrypted, it is impossible for users that do not posses the key to use this navigation system. Whether this is an advantage or disadvantage depends on the application. For military systems this can be an advantage, as enemies will not be able to use these signals.

3.2.2 MAC

Instead of encrypting the data, a message authentication code (MAC) can be used to provide message authentication. In a MAC, the data message and a key are used to generate a tag of a fixed length. It is only possible to generate this tag with the knowledge of the key and the data message. Users with the correct key can use this key and the data message to validate the MAC. If the MAC is correct, the user can be sure that the data message comes from the claimed communication partner. The user can also be sure that the data message has not been changed because changing only one bit of the data will result in a completely different MAC. Without the knowledge of the key an attacker will not be able to generate a valid MAC-data pair.
A MAC is a symmetric security mechanism. This means that the same key is used to create and verify the MAC. Everyone in possession of the key can create valid MACs. Therefore, MACs are only usable in trusted user groups such as the military. The big advantage of a MAC is the rather small size compared to a digital signature. Compared to encryption with an integrity check, the advantage of a MAC is that the data can still be read by users without the correct key. But these users will not be able to authenticate the data.

3.2.3 Digital Signatures

Digital signatures are the asymmetric counterpart of MACs. The big difference between MACs and digital signatures is that a different key is used to create the digital signature and to verify the signature. A secret key is used to create the digital signature for a data message. To validate this signature, a public key is used. If the signature for the data message is valid, the user can be sure that only the entity with the corresponding secret key could have generated the signature. Hence, the user can be sure that the data message really comes from the claimed entity and that it has not been changed. Therefore, digital signatures as well as MACs prevent signal-synthesis and counterfeit-correction message attacks. The big advantage of digital signatures is the asymmetry. The transmitter will use the secret key to sign the data messages. The corresponding public key is available to all users, as attackers can not use them to forge data messages. Therefore, digital signatures can be used without any danger in open communities.
The disadvantage of digital signatures is the size of several hundred bits for one digital signature.

3.3 Hide the signal

A very powerful countermeasure against nearly all attacks is to hide the navigation signal in the noise level. Only the users with the correct key can reveal and use the signal. The best example for this technique is the military GPS P(Y) code. The military Y signal gets multiplied with a secret and very long 10.23 MHz pseudo-random spreading sequence P. This spreads the 100 Hz mainlobe bandwidth of the

data signal by a factor of $2 \cdot 10^5$ to 20 MHz. As a result, the signals peak power-spectral density is reduced by the same factor (53 dB) and ends up roughly 28 dB below the thermal noise density seen by a typical receiver. [13] Hence, in both, the time and frequency domain, the Y signal disappears in the noise. This encrypts the signal similar to a stream cipher. Without the correct spreading sequence P it is not possible to reveal the navigation signal.
Signal-synthesis attacks are impossible, as an attacker will not be able to generate these signals without the secret code. Hidden signals can also prevent signal-synthesis attacks. To successfully launch a selective-delay attack, each navigation signal of the different transmitters needs to be delayed for a different amount of time. To delay each signal for a different amount of time, the signals from all transmitter stations need to be separated from each other. However, if the signals arrive at the same time, this will only be possible if the secret code is known or if the signal can be raised above the noise level. But raising the signals above the noise level is very difficult. It might be possible to raise the P(Y) code above noise with the use of very good high-gain dish antennas with diameters of more than 10 meters. At least four tracking dish antennas or a phased array antenna would be needed to raise four individual signals above noise level. Such an attack would be very complex and very expensive equipment would be needed. The P(Y) codes only repeats itself after several weeks. Hence, an attacker that wants to steal the secret P(Y) code of a satellite needs to be in vision of this particular satellite for several weeks.
The big disadvantage of hidden signals is the need of a symmetric key. If the spreading sequence is known to an attacker, the security of this system would be entirely broken. Therefore, this countermeasure can only be used in trusted user groups such as the military. Otherwise, the secret key might be published. These receivers also need to be tamper proof, as otherwise the secret code might be revealed by tampering a receiver.

3.4 Hidden markers

The idea if hidden markers was first introduced by Kuhn in [13]. Hidden markers are used to prevent selective-delay attacks. The main idea is to hide signals, called hidden markers, in the noise level. These hidden markers can only be recovered using a secret key. This key will be released after some delay d. The user digitizes and buffers the entire antenna input so that the hidden markers and the exact arrival time of the hidden markers are stored. After the delay d, the key will be published and the receiver can reveal the hidden markers and the exact reception time in the recorded noise using this key. If the reception time of the hidden markers match with the navigation signals, the signals are valid. To prevent attackers from creating their own hidden markers with a different key, the key needs to be authenticated, e.g. by using a digital signature. This makes hidden markers an asymmetric security mechanism, as the user only needs to know the public key of the used signature scheme. The user does not need to know any secret information to be able to authenticate the hidden markers. Figure 3.1 illustrates the idea of hidden markers.
To create the hidden markers, direct sequence spread spectrum (DSSS) is proposed

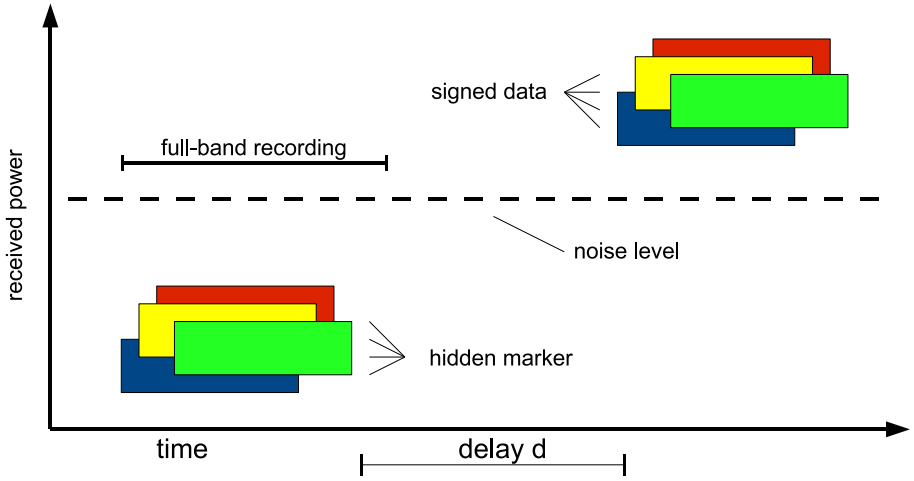


Figure 3.1: The hidden markers from the different satellites are hidden below the noise level. The receiver records the entire bandwidth. After the delay d, the satellites sign and transmit the hidden markers above the noise level. Hence, the receiver can verify the signals by checking the recorded noise for these hidden markers. Each color represents signals from one satellite.

in [13] for satellite based navigation systems, such as GPS, Glonass or Galileo. This is the same technique used by the military GPS P(Y) code to hide the navigation signal in the noise. The security of the system is similar to the security of hidden signals. Without high-gain antennas the attacker will not be able to separate the hidden markers from the noise and from each other, which is necessary to launch a selective-delay attack. The only difference is that after the delay d the spreading sequence is published. This will make it possible for the user to detect the hidden markers in the stored data. But this will also allow an attacker to create hidden markers on his own using this code. However, the attack can only be launched with a delay, which will create a clock offset of at least d. If d is large enough, this offset can be detected using low-cost crystal oscillators.
The spreading code needs to be authenticated so that an attacker can not simply use his own code. This can be done using the methods described in section 3.2. The other alternative is to use a self authenticated one-way chain. These chains are described in section 5.2.
It is important that the hidden markers from different stations arrive at the receiver at the same time. Otherwise an attacker can perform a selective-delay attack by delaying the entire noise in a way that each hidden marker will be delayed according to the attack scenario. To make sure that the navigation messages match after such a selective-delay attack, the attacker needs to be able to remove the original navigation messages from the noise. If the hidden markers arrive during the same time period, a selective-delay attack becomes impossible, because each hidden marker can not be delayed with an individual offset.
Relaying attacks against hidden markers are possible if the attacker is able to relay

the navigation signals as well as the entire noise. However, hidden markers increase the complexity of the attack, as the entire noise needs to be relayed. Without the hidden markers, an attacker might only need to relay the navigation messages. No method that is resistant against relaying attacks for broadcast navigation system is known today.
One disadvantage of hidden markers is the high storage requirements for the receivers to store the noise. In [13] Kuhn estimates for his implementation of hidden markers for GPS that a receiver needs to store about 25 MB data for a hidden marker of a length of 1 second. However, it is probably possible to use hidden markers much smaller than 1 second to decrease the storage requirement.
The other drawback is that for some positioning systems a realization of hidden markers is very difficult. Using DSSS requires to transmit the signal in a much higher frequency than the original data signal. Therefore, the use of DSSS in low-frequency systems such as LORAN is very restricted. Terrestrial positioning systems such as LORAN also have to face the problem that the users and the potential attackers can be as far away as several hundred kilometers but can also be as close as several meters from the transmitter station. To add a hidden marker into the noise level that can be validated by a receiver which is 600 kilometers away but is still hidden for an attacker only hundred meters away from the transmitter is a very challenging task. Other techniques to hide the signal might be needed to realize hidden markers in terrestrial positioning systems.
The big advantage of hidden markers compared to hidden signals is that it is an asymmetric security mechanism. The user does not need to possess a secret to be able to verify the location. If one receiver gets tampered, this will have no impact on other receivers as no receiver possesses a secret.

3.5 Countermeasures for active positioning systems

The countermeasures above describe the different methods known today to secure passive positioning systems such as LORAN or GPS. In passive positioning systems, the user only receives data but does not send out data himself. In active positioning systems, the user sends out navigation messages to the transmitter stations as well. In these systems, other powerful mechanisms are known to secure the location determination. As the user has to communicate with the stations, these systems are mostly limited to small coverage areas and are not useful for global positioning. The most powerful countermeasure in these systems is distance-bounding. In distance-bounding, the base station sends out a challenge to a user. The user sends back the answer of the challenge. The station measures the time it took the user to answer the challenge. Typically, radio waves are used as the communication channel. As the station knows the speed of the radio waves, the station can use the answer time to calculate an upper bound of the distance between the user and the station. The user repeats this challenge and response system with at least two other stations. In this way, the stations estimate an upper bound of the distance between the user and the different stations. The stations exchange this information with each other and determine the exact two-dimensional position of the user.

The attacker can only delay the response and hence increase the calculated distance between a station and the user. However, the attacker will not be able to answer quicker to the challenge and hence will not be able to pretend to be closer to a station than the user actually is. But if the user pretends to be farther away from one station than he actually is, he also needs to pretend to be closer to another station if the user is within a triangle defined by three stations. As the attacker can not pretend to be closer to a station, this attack will be detected. Figure 3.2 illustrates this idea.
As eLoran is a passive positioning system, the user cannot communicate with the transmitter station. Therefore, distance-bounding or other countermeasures that need communication from the user to the transmitter stations can not be used to increase the security of eLoran.

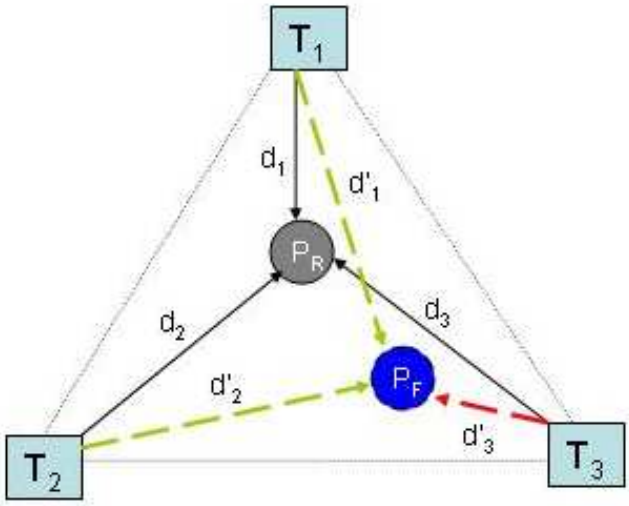


Figure 3.2: Using distance bounding to prevent spoofing attacks. The transmitter stations T_1, T_2 and T_3 use distance bounding to set a lower bound of the distances d_1, d_2 and d_3 between the transmitters and the user's position P_R. An attacker will not be able to pretend to be at location P_F, as he can only delay messages and thereby increase the distances d'_1, d'_2 and d'_3 between the attacker and the transmitter. However, d'_3 needs to be smaller than d_3 and therefore transmitter T_3 will detect the attack. (The coverage area of this system is given by the triangle T_1, T_2 and T_3.)

4 eLoran and the LORAN Data Channel

4.1 LORAN

LORAN is a terrestrial positioning system build in the 50s by the USA. It operates with high power and in the low-frequency band of 100kHz. Due to the low frequency and the high power, the signals have a long range and users at distances of 800 km or more can receive these signals. As the signals are very strong, they can reach places that are not reached by GPS, like urban areas and even indoors. The time difference of arrival (TDOA) of the signals is used to determine the position. The LORAN stations are divided into chains. Each chain covers a certain geographic area and consists of one master station and at least two secondary stations, also called slaves. As all stations transmit on the same carrier frequency, time-division-multiple-access is used to avoid interference. According to a group repetition interval (GRI), the master station transmits a LORAN pulse group, containing of 9 LORAN pulses. Each secondary station responds with a LORAN pulse group, containing 8 LORAN pulses, after an individual time interval, called the emission delay. The emission delay is chosen in a way such that two signals of one chain do not collide within the defined coverage area of the LORAN chain. The GRI defines how long the master waits until it starts another repetition by sending out the next group of pulses. The GRI is unique for each LORAN chain and differs between 0.05930 seconds for the Canadian East Coast Chain to 0.09990 seconds for the North Pacific Chain. The different GRIs are used to identify the LORAN chains. The master station used to be identified by the transmission of the additional ninth pulse. But nowadays, the master station is usually identified by the phase coding described in section 4.1.1. A GRI is denoted in increments of ten microseconds. Hence, the West Coast 9940 LORAN chain has a GRI of 0.09940 seconds. Some stations are used as secondary stations in two chains. These stations are called dual rated and transmit pulses for both chains.

The position is computed by using differences in pulse group arrival times between the master station and the secondary stations. A hyperbolic line defines the positions that have the same time difference of arrivals. (see figure 4.1) Using the time difference of arrivals between the master station and another secondary station, a second hyperbolic line is defined. The intersection point of these two lines determine the two dimensional position of the user. As only the time difference of arrival is used in LORAN, and not the time of arrival of a particular signal, time synchronization is not required.

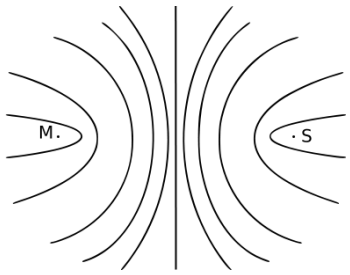


Figure 4.1: The lines between M and S are defined by different time differences of arrivals.

4.1.1 LORAN pulse

One pulse group transmitted by the stations consists of either 8 or 9 LORAN pulses. A LORAN pulse can be seen in figure 4.2 and is given by following equation:

$$p(t) = (t-\tau)^2 e^{\frac{-2(t-\tau)}{65\mu sec}} sin(2\pi \frac{t}{10\mu sec})$$

where $(t-\tau)^2 e^{\frac{-2(t-\tau)}{65\mu sec}}$ represents the envelope of the signal and $sin(2\pi \frac{t}{10\mu sec})$ defines the carrier. LORAN pulses have an initial phase shift of 0 or 180 degrees, which is denoted with a + or -. These phase shifts are known as phase coding and repeat every two GRIs. The first GRI is denoted with A and the second with B. The phase coding of the master station and of the secondary station differ from each other, so that the phase coding can be used to identify the master station. Table 4.1 shows the LORAN phase codes.

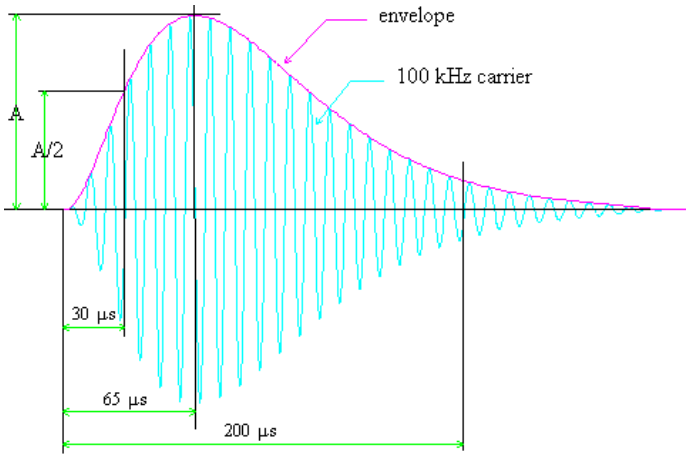


Figure 4.2: A single LORAN pulse.

Phase coding is used in LORAN to mitigate the effects of interference from other undesired LORAN pulses such as sky wave interference and signals from other LORAN chains. The goal of the phase code is that if the signal is processed over any two GRI intervals, undesired LORAN pulses should average to zero. The reason

GRI Interval	Master Station	Secondary Station
A	++–+-+- +	+++++–+
B	+–+++++ -	+-+-++–

Table 4.1: LORAN Phase Code

for this is that the auto-correlation of two GRI sequences results in a value of zero except for the case that the interfering signal has the same phase code and an offset of less than one pulse length. [17]
The exact time of arrival of a LORAN pulse, as it is needed for positioning, is generally defined as the sixth zero crossing of the carrier in each pulse. Typically, the arrival time of one LORAN pulse group is determined by averaging the arrival time of each of the eight LORAN pulses.

4.1.2 Interference in LORAN

The main source of interference in LORAN is skywave interference. The LORAN signals propagate either as a ground wave along the earth's surface or as a sky wave by reflecting from the ionosphere. The ground waves are used for the position calculation as they are more stable and reliable. Sky wave interference is caused by signals that are reflected back to the ground by the ionosphere. As these signals travel longer than the signals of the ground wave, they arrive with a delay. These skywave signals can be stronger than the ground wave signals and can cause big interference problems. The arrival delay and signal strength of sky waves can differ greatly, as it depends on the height of the ionosphere, time of day and the number of times the signal reflects from the ionosphere. Phase coding is an efficient way to minimize the effect caused by interference from sky waves with a long delay. The biggest problem is interference caused by sky waves with very short delays. Different sky wave models can be used to reduce this interference.
Each LORAN chain is designed in a way that two ground wave signals from two different stations within a LORAN chain do not interfere with each other within the coverage area of the chain. However, signals transmitted by LORAN stations from other chains can create interference. This kind of interference is called cross rate interference. Each LORAN chain has its own individual GRI and the cross rate interference can be computed in advance using a model for the ground wave propagation and signal strength. Because of the different GRIs, the cross rate interference changes in time so that it is ensured that cross rate interference only appears temporarily at one location.
Lightnings can also create interference, although the biggest interference is caused by sky wave interference and cross rate interference. [17] Other radio sources, such as unintentional transmission on 100 kHz, are normally too weak to cause interference problems for LORAN.

4.2 enhanced LORAN

The old LORAN-C system does not provide the increased performance requirements of modern positioning systems. Therefore, an updated version of LORAN, called enhanced LORAN (eLoran) is being developed by the International LORAN Association. This enhanced version of LORAN will be able to meet the accuracy, availability, integrity and continuity performance requirements needed for aviation non-precision instrument approaches, maritime harbor entrance and approach maneuvers, land-mobile vehicle navigation and location-based-services. It will also be a precise source of time and frequency that meets the Stratum 1 requirements. [1] The performance improvements will enable eLoran to be an independent backup for GNSS systems. The eLoran system is still under development and has not been defined completely. In the following, eLoran is introduced based on the Enhanced LORAN (eLoran) Definition Document Version 1.0 published in October 16th 2007. [1] As the definition of eLoran is an ongoing project, the eLoran design might still change in the future. The eLoran system performance requirements are listed below. [1] In addition to the performance requirements, eLoran is required to be backward compatible with old LORAN-C receivers.

Accuracy is defined as the difference between the position estimated by the receiver and the true position of the receiver which is only exceeded in 5% of the time in the absence of system failures. The accuarcy requirements for eLoran are 8-20 meters.

Availability is the fraction of time for which the system is operational. For eLoran a availability of 99.9%-99.99% is required.

Integrity The portion of time the system exceeds the alert limits of the system without raising an alarm. The eLoran requirement for integrity is that the alert limits are only exceeded in one second out of 10^7 seconds without raising an alarm.

Continuity The chance that an alarm is raised during the time of an operation. A continuity of 0.999 over 150 seconds is required, which means that with a chance of at least 99.9% no alarm will be raised during a 150 seconds periode. (Whether this alram is a false alarm or not does not matter)

There are three main changes in eLoran compared to LORAN-C, a data channel, TOA, and correction messages. First of all, eLoran will contain a data channel for navigation and time messages. These time informations will enable the user to use Time-Of-Arrival (TOA) instead of Time-Difference-Of-Arrival (TDOA). In time-of-arrival (TOA), the arrival time of each signal is used to compute the pseudo-range between the receiver and the transmitter stations. Using the pseudo-ranges of three different transmitter stations, the exact two-dimensional position and time can be computed. The difference between TOA and TDOA is that in TOA every transmitter station can be used for positioning, while in TDOA only the stations within the same chain can be used. The TDOA system limits LORAN to use only signals from

the same chain, even if strong and clear signals from other chains can be received. TOA will enable the receiver to use these signals as well. This all-in-view mode will help to get the most accurate and reliable position and timing out of the available signals.
The third change that will ensure great performance improvements is the transmission of real-time correction and warning messages. A network of monitor sites and reference stations within the eLoran coverage area will monitor the performance of eLoran and will provide warnings in real time if abnormalities are detected. Furthermore, these reference stations will generate correction messages that are transmitted over the LORAN Data channel. These correction messages will significantly increase the performance of eLoran.
On the hardware side, the transmitter stations are being updated to meet the higher performance requirements. All eLoran transmitters will use modern solid-state transmitters (SSC) and control technologies. Uninterruptible power supplies (UPS) will ensure greater availability and reduce the risk of interference caused by a failure of the incoming power. Furthermore, multiple modern cesium clocks will provide accurate timing, completely independent from GPS. The time information provided by eLoran will enable the user to synchronize to UCT time with an accuracy of 50 nanoseconds.
These changes will make eLoran meet the advanced requirements of todays PNT systems and will enable eLoran to be a completely independent backup for current GNSS systems.

4.3 LORAN Data Channel (LDC)

The LORAN Data Channel is the key improvement in eLoran. It will carry the time information needed to switch from time difference of arrival (TDOA) to time of arrival (TOA). It will also carry correction messages generated by the reference stations needed to meet the performance requirements. Furthermore, it will provide almanac data, containing all important information about the reference and transmitter stations and the current state of eLoran. The definition of the data messages is still an ongoing process and new message types might be included in the future.
The data is modulated by inserting a ninth-pulse to the secondary stations. In LORAN-C, each secondary LORAN pulse group consists of eight LORAN pulses. In eLoran, each secondary station will transmit an additional ninth LORAN pulse per pulse group. This ninth pulse carries the data by using pulse-position modulation. The delay between the eight pulse and the ninth pulse contains the data information. 32 different states are defined for the position of the ninth pulse, so that 5 bits can be transmitted by each pulse group. This data modulation technique is called ninth-pulse modulation. The ninth pulse is transmitted 1000 microseconds after the 8th pulse, plus the individual delay for each symbol. The ideal delays for each symbol in microseconds are given by the formula:

$$d_i = 1.25 \cdot (i \bmod 8) + 50.625 \cdot \lfloor i/8 \rfloor$$

The actual delay values are shifted to coincide with the ticks of a 5MHz clock.

Table 4.2 lists the individual delays for each symbol.

i	μs	i	μs	i	μs	i	μs
0	0	**8**	50.6	**16**	101.2	**24**	151.8
1	1.2	**9**	51.8	**17**	102.6	**25**	153.2
2	2.6	**10**	53.2	**18**	103.8	**26**	154.4
3	3.8	**11**	54.4	**19**	105	**27**	155.6
4	5	**12**	55.6	**20**	106.2	**28**	156.8
5	6.2	**13**	56.8	**21**	107.6	**29**	158.2
6	7.6	**14**	58.2	**22**	108.8	**30**	159.4
7	8.8	**15**	59.4	**23**	110	**31**	160.6

Table 4.2: Symbol delays from zero-symbol offset in microseconds

LORAN messages:
One LORAN message is 120 bits long. To transmit these 120 bits, 24 pulse groups, each containing a 5-bit symbol, are needed. The transmission time of the 24 pulse groups depends on the Group Repetition Interval (GRI). The maximal time needed to transmit the 120 bits will be about 2.4 seconds for the North Pacific Chain with a GRI of 9990.
Each message consists of a 4-bit header, a 41-bit payload, and a 75 bit parity component. The four bit header specifies what type of LORAN message is being sent. There are 16 different message types possible, but so far only fife types have been defined.
The LORAN data channel has to face a lot of interference, especially due to cross-rate interference. The Reed-Solomon-Forward-Error-Correction is used to ensure the reliable reception of the data messages. The Reed-Solomon-Forward-Error-Correction code (RS) was designed for correcting burst errors. The data message is divided into blocks of symbols with a fixed length. In eLoran 5-bit symbols are used as the ninth-pulse modulation is design to transmit 5-bits with one signal. For eLoran RS(31,16) is used with 31 code symbols of which are 9 data symbols, 15 are parity symbols and 7 are padded with zero. The 7 zero symbols are not transmitted and are only used to compute the parity symbols. The 9 data symbols and the 15 parity symbols are further encoded by adding a coset vector of 24 5-bit words to the symbol. This is done to eliminate cyclic problems with the Reed Solomon code.[2]
The Coset Vector = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23] is simply added modulo 32.
The used RS code can correct up to 15-9+1=7 symbol errors. As each symbol consists of 5-bit, the RS code can correct up to 7 5-bit burst errors. The biggest interference problems the LORAN data channel has to face are data losses due to cross-rate interference. These burst errors are predictable and can be eliminated using the Reed-Solomon coding. However, this comes with the cost of sending 75 parity bits for 45 data bits.

The exact design of the navigation messages is still under development. I will only go into detail of the proposed time and station identification message. The Time-of-day message contains the absolute time expressed as the number of seconds since Jan. 1st 1958 00:00 until the time of transmission of the first pulse of the first GRI of this message. A receiver can use this information to calculate the exact transmission time of the first pulse within nanosecond accuracy. The exact transmission time can be calculated using following formula:

$$T = \lceil numberSeconds/GRI \rceil \cdot GRI + ED$$

where *numberSeconds* denotes the number of seconds since Jan 1st 1958 00:00 and the transmission of the first pulse of the first GRI containing the time information. ED is the published emission delay of the transmitter station sending this message and T is the exact time since Jan 1st 1958 00:00.
The time-of-day message also includes a station ID to identify each of the eLoran stations. Besides the time message, correction messages generated by the reference stations will be transmitted. An almanac message will be sent out containing information of all eLoran stations and the current status of eLoran. For example, the almanac will contain the emission delay ED for every eLoran transmitter. Other messages might be implemented in the future but no information about these messages is available yet. In this thesis, I suggest to add an authentication message to increase the security of eLoran. (see chapter 6.1)

4.4 Over-the-air attacks against LORAN

In chapter 3, different attacks on positioning systems have been introduced. It has been mentioned that over-the-air attacks on LORAN are much harder than over-the-air attacks on satellite based system such as GPS. In this section, concrete results, on how much effort is needed to attack LORAN, are presented. These results are based on an analysis done by Lo, published in [18]. The problem an attacker faces when he tries to attack LORAN is that LORAN operates in the low frequency band and has transmission powers of 400kW or more. At 100kHz, the signal has a wavelength of three kilometers. How efficiently a radio signal can be transmitted depends on the ratio of the wavelength compared to the transmitter antenna. A rule of thumb says that the antenna efficiency is proportional to the square of the antenna height. With a wavelength of 3km, even a quarter wavelength antenna is extremely hard to realize. Common LORAN antennas are 190m top loaded monopole antennas, but antennas as tall as 411.48m have been used. To jam a LORAN signal you basically have to overcome the broadcast power of the signal. This can be done by transmitting roughly the same received power at the same carrier wave frequency. The power of a LORAN signal falls of greater than the nominal square strength of the distance because of the attenuation for propagation along the ground. A 400kW transmitter 300 km away is roughly equivalent to a 40 W transmitter 5km away or a 4W transmitter 0.5 km away. The 300km assumption was chosen as a reasonable distance between a user and a close transmitter.

Spoofing a LORAN signal is even more challenging than jamming. Transmitting a LORAN signal from a short, high Q antenna creates a signal with a much narrower signal bandwidth than the original LORAN signal. Therefore, the transmission of a LORAN pulse on a short antenna is even more inefficient than the transmission of a pure tone at a frequency of 100kHz. For jamming, the attacker does not need to transmit a LORAN pulse but can simply transmit a pure tone.
Besides generating a new LORAN pulse for spoofing, the attacker can also use the design of the LORAN signal to introduce a small position error while using much less energy. The arrival time of a LORAN pulse is defined as the sixth zero crossing of the LORAN pulse. By overlaying a pure tone on the actual LORAN signal, this tracking point can be shifted, resulting in a false computed arrival time at the receiver. The advantage of this attack is that it requires much less energy than overcoming the original signal. The disadvantage of this method is that only an error much smaller than the LORAN wavelength of 3km can be introduced. Another huge disadvantage is that there are several ways how the attack can be detected. The overlayed signal will create changes in the signal envelope. These changes will have the effect that different errors occur at different tracking points. For example, a spoofing error of 239m at the sixth zero crossing tracking point results in a 280m error at the 5th zero crossing and a 340m error at the 4th zero crossing. [18] By using multiple tracking points to calculate the position and comparing the results with each other, this type of spoofing attack can be detected. Furthermore, to be able to introduce the wanted error to the tracking point, the exact transmission time of the signal needs to be known in advance. As the transmission times of the LORAN pulses are public and predictable, this is true for the LORAN pulses. In eLoran, the transmission time of the additional ninth pulse depends on the transmitted data. If this data is not known to the attacker (e.g. part of an authentication message is being transmitted), the attacker has to guess the transmission time of the ninth pulse. If this guess is not correct, this will either result in a data bit error or will change the shape of the ninth pulse in a way that the receiver gets suspicious. In this way, pulse position modulation adds to the security of eLoran. Furthermore, message authentication, as it is proposed in chapter 6, will also add to the security against this attack, as the message authentication data is not predictable in comparison to time or correction messages.
According to Lo's calculations, an attacker needs 160mW to introduce a 30m error if he is 5km away from the receiver. To introduce a 150m error 4W are needed. On the first sight, these values seem very low and easy to achieve. However, transmitting this power on a 100kHz frequency with small antennas is very challenging. The antenna size an attacker could use is very limited in practice. It is very unrealistic that an attacker is able to operate a 190m antenna 5km away from the victim's receiver. The attacker does not want to be detected, so that the attacker needs to be able to quickly set up the antenna or needs to be able to set up the antenna without raising suspicion. In the following, the needed antenna height for different attack scenarios is analyzed. We assume that the attacker uses a simple monopole antenna that is reasonable short of about 30m or less. Other antenna structures such as guy wires may provide better results. However, these structures would

require significantly more set up time and costs. Furthermore, the assumptions in this analysis represent an optimistic case from the attacker's point of view, as many losses such as ohmic and matching losses, as well as transmitter inefficiencies are omitted. It also assumes a perfect ground plane which will be very hard for an attacker to achieve.

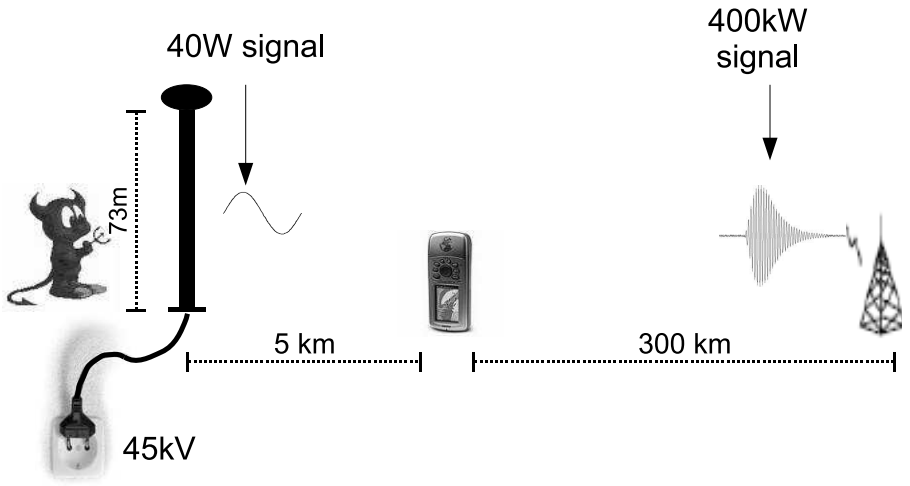


Figure 4.3: Illustration of a jamming attack on LORAN. The victim's receiver is 300km away from the LORAN station that has a transmission power of 400kW. An attacker 5 km away needs 40W transmission power to overcome the original LORAN signal. The attacker would need a ca. 73 meter monopole antenna with a radius of 50 mm and 45kV to create the needed 40W transmission power in the used 100kHz band.

The maximal voltage potential available to the attacker is assumed to be 45kV. This is a rather reasonable maximum, as it will be very difficult for the attacker to have access to more power than 45kV and still being inconspicuous and undetected. With these assumptions, table 4.3 shows the needed antenna heights for the different attack scenarios. If the transmitter station is 300km and the attacker

Scenario (5km, 0.5km)	$a = 2.3$ mm	$a = 25.4$ mm	$a = 50$ mm
Jamming (40W, 0.4W)	90m, 27m	78m, 22m	73m, 21m
Spoof 30m error (160mW, 1.6 mW)	21m, 6.1m	17m, 4.7m	16m, 4.2m
Spoof 150m error (4W, 40mW)	49m, 14m	42m, 12m	39m, 11m

Table 4.3: Required monopole antenna heights for the different attack scenarios and different antenna radii a if the attacker is either 5km or 0.5km away from the victim's receiver.

5km away from the receiver, jamming a LORAN signal requires a 73m monopole antenna with a radius of 50mm and 45kV. This attack is illustrated in figure 4.4. It is extremely difficult to realize such an attack and it is even much more unrealistic that the attacker stays undetected. It should be very easy to detect a 73m

antenna transmitting such a strong signal within a 5km radius. Jamming a receiver 500m away from the attacker still requires an antenna height of 21m. Detecting a 21 meter antenna within a radius of 500 meters should be very easy as well. On the other hand, setting up a 21 meter antenna and operating it with 45kV will very likely exceed the capabilities of an attacker. Attacks such as selective-delay attacks require even more power, making these attacks even more unlikely.
A spoofing attack, by overlaying a pure tone on the LORAN pulse, requires less power and therefore less antenna heights. These attacks can be detected, but the receiver might not be able to compute a position as accurate as without this attack. So the user might know that his position is being compromised, but might not be able to determine the exact location. The complexity of the attack on the other hand is still very big. Even operating a 4.2 meter antenna with 45kV is a challenging task for an attacker, especially if the attacker wants to stay undetected.

This analysis shows why eLoran is much more secure against over-the-air attacks than GNSS systems. In GNSS systems, over-the-air attacks can easily be done using portable, battery operated devices. Furthermore, it can be very difficult to locate a GNSS jamming device, which an incident in the Moss Landing Harbor showed. It took several month to locate the source of interference that caused GPS jamming within the harbor area.[5] Using an array antenna, it is very easy to locate a LORAN jammer, especially keeping in mind that the jammer's antenna has the size of dozens of meters. Hence, even without any security mechanisms, eLoran will provide a much more secure positioning system compared to the civil GPS.

The technique of shifting the tracking point by overlaying a pure tone could be used to launch a counterfeit correction message attack against the LORAN data channel. But such an attack would be very difficult. The reason for this is that the tracking point can only be shifted for a fraction of the wavelength. The wavelength of LORAN is 3km or 10 microseconds. Hence, the signal can only be shifted for a few microseconds. Shifting the signals for such a small amount of time will only introduce a one or two bit error to the 5-bit symbol. However, as the Reed-Solomon forward error correction is used, even a one bit change of one data symbol results in several changes of the parity symbols. Several of these party symbols will very likely differ in more than one or two bits. This makes counterfeit-correction message attacks on eLoran very difficult. Furthermore it is possible to check the different tracking points of the ninth signals for contradictions, like it can be done for the normal navigation signals. Because big time differences are needed to spoof a data bit, the differences in each tracking point will be very big. Therefore, it is very easy to detect this type of attack against the LORAN data channel.

5 Authentication methods for the LORAN Data Channel

The big disadvantage of the LORAN Data Channel (LDC) is its small data rate of only 17 bits/second. The LDC is used to transmit time and station information as well as correction messages. These messages must be sent frequently to meet the strict performance requirements necessary to be a backup for GPS. [22] This leaves only a small portion of the already very low data rate for authentication purposes. Therefore, an authentication mechanism for eLoran with a signature size as small as possible is needed.

5.1 DLP based signature schemes

The most common digital signature schemes are based on the discrete logarithm problem (DLP). For a finite group G, to find a $x \in N$ for a given $g \in G$ and $y \in G$ with $y = g^x$ is called the discrete logarithm problem. In the next section, two different DLP-based signature schemes with relatively short signatures are introduced.

5.1.1 DSA

The Digital Signature Algorithm (DSA) was proposed in 1991 by the National Institute of Standards and Technology (NIST) and became the U.S. Federal Information Processing Standard (FIPS 186) called the Digital Signature Standard (DSS). The DSA is based on the ElGamal signature scheme.

Key generation [20]

1. Select a prime number q where $2^{159} < q < 2^{160}$.
2. Select a prime number p where $2^{1023} < p < 2^{1024}$, with the property that q divides $(p-1)$.
3. Select a generator α of the unique cyclic group of order q in $\mathbb{Z}_p^*$.
4. Compute $\alpha = g^{(p-1)/2} \bmod p$.
5. Select a random integer a such that $1 < a < q-1$
6. Compute $y = \alpha^a \bmod p$

The public key consists of p, q, α and y. The private key is a.

Signature generation

1. Select a random secret integer k, $0 < k < q$ and $gcd(k, q) = 1$.

2. Compute $r = (\alpha^k \bmod p) \bmod q$
3. Compute $k^{-1} \bmod q$.
4. Compute $s = k^{-1}(h(m) + ar) \bmod q$

The signature for the message m is the pair $(r,\ s)$. $h()$ denotes a hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.

Signature verification

Given is the public key $(p,\ q,\ \alpha,\ y)$, a message m and the signature $(r,\ s)$

1. Verify that $0 < r < q$ and $0 < s < q$
2. Compute $w = s^{-1} \bmod q$ and $h(m)$.
3. Compute $u_1 = w \cdot h(m) \bmod q$ and $u_2 = rw \bmod q$.
4. Compute $v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$.

The signature is valid if $v = r$.

Security and signature size

There are two distinct but related discrete logarithm problems in the DSA. One is the logarithm problem in $\mathbb{Z}_p^*$, where the powerful index-calculus methods can be used. The other is the logarithm problem in the cyclic subgroup of order q, where the best current methods run in square-root time, $O(\sqrt{q})$. [20] As the length of the signature depends on q and not on p, p can be chosen sufficiently long without having impact on the signature size. For a security equivalent to a 80 bit symmetric cipher, q needs to be at least 160 bits. As the signature consists of two numbers of the subgroup of order q, this results in a signature size of at least 320 bits. However, it is believed that DSA with q of the size of 160 bits will not be secure for long term applications. A DSA signature with an approximate equivalent security as a 120 bit block cipher has a length of about $2 \cdot 240$ bits = 480 bits.

5.1.2 ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) is an ANIS, IEEE and NIST standard for digital signatures. It is the elliptic curve analogue of the DSA. Describing the details of elliptic curve cryptography would go beyond the scope of this work. Only a very brief introduction of the functionality of ECDSA is given in this paper. The big advantage of the ECDSA is that no subexponential-time algorithm is known for the elliptic curve discrete logarithm problem. [11] Therefore, the subgroup can be much shorter compared to signature schemes based on other discrete logarithm problems. This not only results in great performance achievements, but also in a short signature.

Elliptic curves over $\mathbb{F}_p$

Let $p > 3$ be an odd prime. An elliptic curve E over $\mathbb{F}_p$ is defined by an equation of the form $y^2 = x^3 + ax + b$, where $a, b \in 4a^3 + 27b^2 \not\equiv 0 \pmod p$. The set $E(\mathbb{F}_p)$ consists of all points $(x, y), x \in \mathbb{F}_p,\ y \in \mathbb{F}_p$, which satisfy the equation $y^2 = x^3 + ax + b$,

together with a special point $\mathbb{O}$ called the point at infinity.
The group operation is point addition, $P_1+P_2 = P_3$, and point doubling, $2P = P+P$. Another important arithmetic is the scalar point multiplication, an operation of the form $k \cdot P = Q$ where k is a positive integer and P and Q are points of the elliptic curve. The scalar point multiplication is defined as adding k times the point P to itself. So $k \cdot P = \sum_1^k P = Q$. The inverse operation of the scalar multiplication, so the operation to recover k for a given P and Q with $k \cdot P = Q$, is called the elliptic curve discrete logarithm problem. Today, no subexponential-time algorithm is known for this problem.[11]

ECDSA Key generation

Given are the domain parameters for the elliptic curve E and a Point $P \in E$ of the order n.

1. Select a random or pseudo random integer d where $1 < d < n - 1$.
2. Compute $Q = dP$.

The public key is Q and the private key is d.

ECDSA signature generation

Given is the public point $P \in E$ of order n, the private key d and the message m.

1. Select a k with $i < k < n - 1$.
2. Compute $kP = (x_1, y_1)$ and convert x_1 to an integer $\widehat{x_1}$.
3. Compute $r = \widehat{x_1} \bmod n$. If $r = 0$ then repeat from step 1.
4. Compute $e = h(m)$.
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$ then repeat from step 1.

The signature consists of the two values r and s.

ECDSA signature verification

Given is the public point $P \in E$ of order n, the public key Q, the message m and the signature (r, s).

1. Verify if $r, s \in [1, n - 1]$.
2. Compute $e = h(m)$.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $X = u_1P + u_2Q = (x1, y1)$.
6. If $X = O$ then Return("Reject signature");
7. Convert x_1 to an integer $\widehat{x_1}$ and compute $v = \widehat{x_1} \bmod n$.

If $v = r$ then the signature is valid, otherwise the signature is invalid.

Security and signature size

Elliptic curve cryptography was first suggested independently in 1986 by Miller [21] and Koblitz [12]. In the last twenty years it has been the focus of a lot of research, but no weakness has been discovered so far. The security of the ECDSA is

based on the elliptic curve discrete logarithm problem. The best known method for solving the elliptic curve discrete logarithm problem is the (parallelized) Pollard's Rho Algorithm. The runtime of the Pollard's Rho Algorithm is about $O(\sqrt{n})$, where n denotes the order of the point P. For a security equivalent to a 80 bits symmetric cipher (also written as O(80)), n needs to be about 160 bits long. Hence, the signature size for O(80) is about $2 \cdot 160$ bits =320 bits. Therefore, the signature is as long as the signature of the DSA. The advantage of ECDSA compared to DSA is that the computation cost during the signing and verification process is much lower.

5.2 TESLA

In the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) signature scheme, only symmetric algorithms are used. To achieve the needed asymmetry, TESLA uses time. The basic idea of TESLA is to authenticate the messages with MACs and to reveal the corresponding keys after some delay d. An entity will not be able to authenticate the message until it receives the corresponding key. After the key is published, the receiver can use this key to verify the MAC and the corresponding message.
Obviously, after the key has been published, an attacker could use this key to generate a valid MAC. However, a receiver only accepts MACs before the corresponding key has been released. To be able to reject old MACs, the transmitter and the receiver need to have at least loosely synchronized clocks. To be more precisely, the maximum offset between the receiver's clock and the transmitter's clock needs to be smaller than the delay d.
The key for each MAC is authenticated using a self authenticating one-way chain. In the following, the generation of the one-way chain and how this one-way chain can be used to authenticate keys is discussed.

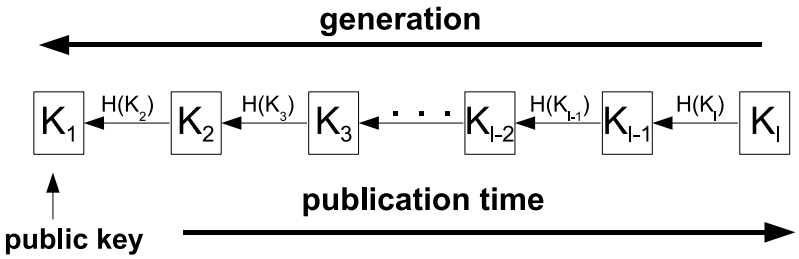


Figure 5.1: The generation and publication of a TESLA one-way key chain.

Key-chain generation:

To build a one-way chain, a one-way function H is needed. In a one-way function, the computation of an output $y = H(x)$ should be easy, while the inverse computation of a value x for a given y should be computationally infeasible. In practice, hash functions are often used, so that one-way chains are often called hash chains.

To generate a one-way chain of length l, at first a random number r is chosen. The first node K_l is computed using the random number and the one-way function H(), with $K_l = H(r)$. Then the rest of the nodes are generated by computing $K_i = H(K_{i+1})$ for $i = l - 1$ to $i = 1$.
In TESLA, the private keys are the nodes of the one-way chain K_l,...,K_2. The public key is the last computed key K_1 and the time intervals during which each key is valid. The transmitter reveals the keys in the reverse order of the computation of the key. So the key that will be released first is K_2, the next key is K_3 and the last one will be K_l. When a receiver receives a key K_i at time T_i, the receiver can validate the key by comparing the output of $H(K_i)^{i-1}$ with the public key K_1. If these values are the same and the time interval for key K_i matches the current time, the key is valid. The one-way chain generation and publication is visualized in figure 5.1.

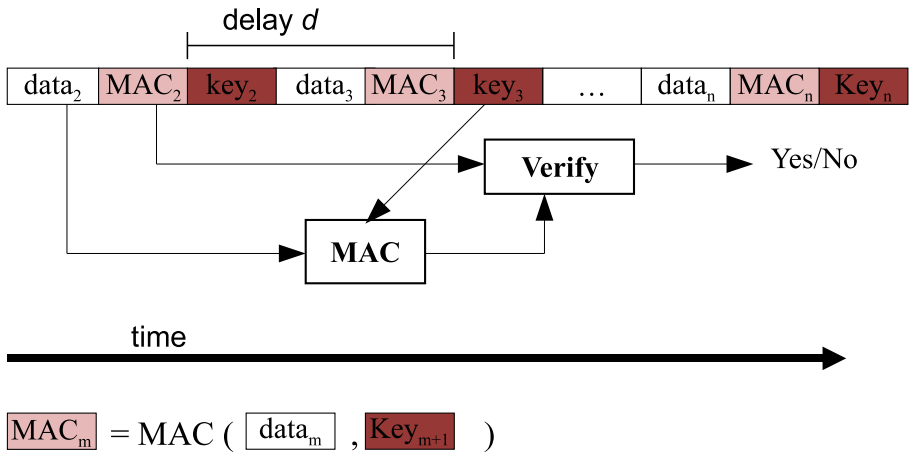


Figure 5.2: The basic structure of TESLA.

Signature generation:

To sign a message m_i during the time interval T_i, the transmitter generates a MAC $MAC_{K_{i+1}}(m_i)$ with the key K_{i+1}, which will be revealed at some time after the current time interval T_i. During the time interval T_i, the transmitter sends out the message m_i, the MAC $MAC_{K_{i+1}}(m_i)$ and the key K_i which is supposed to be released during this time interval. The receiver cannot authenticate the message m_i yet, because he does not know the key K_{i+1}. During the time interval T_{i+1}, the transmitter sends the current message m_{i+1}, the MAC $MAC_{K_{i+2}}(m_{i+1})$ and the key K_{i+1}. The receiver now validates the key K_{i+1} by computing $K_1 = H^i(K_{i+1})$ and comparing this value with the public key. After the receiver has verified the key, the receiver can verify the authentication of the message m_i by computing $MAC_{K_{i+1}}(m_i)$. Hence, the receiver can only authenticate messages after some delay d. This delay d is defined by the time until the needed key is revealed. This delay is also a security parameter, as it defines the allowed maximal offset between the clock of the receiver and the transmitter. Figure 5.2 illustrates the verifying process.

5.3 Adjusted TESLA

Traditionally, TESLA is used to authenticate data messages. However, the content of the data messages is not the most important part in positioning systems. In some cases, this data only provides correction data and time information that might not be essential for the position determination. Message authentication is needed to defend against counterfeit correction message attacks, but it is not necessary to have message authentication to prevent signal-synthesis attacks. To protect against signal-synthesis attacks, it only needs to be ensured that the signals can not be generated by an attacker. If the attacker can generate parts of the signal, but not all needed parts of the signal, a receiver does not accept this signal. In most systems, only two information are needed to determine a position, the source and transmission time of the signal. If the receiver can be sure that the received signals come from the transmitter station and knows the exact transmission time, the receiver can securely compute the position without the need of additional information. To prevent a signal-synthesis attack, the signal needs to be somehow unpredictable for an attacker so that an attacker can not generate valid signals.
Transmitting a valid key from the one-way chain can be used as source authentication. Based on the design principles of one-way chains, no unauthorized entity should be able to generate valid keys for a one-way chain. Hence, only the valid transmitter is able to reveal a key. The only option an attacker has to be able to transmit a valid key is to replay an old key. As one requirement of TESLA, the clocks of the receiver and the transmitter need to be loosely synchronized. If the attacker uses a key older than the delay d, the receiver will detect that the key is not valid. However, an attacker can delay current keys with a delay which is small enough such that the receiver does not get suspicious. Such an attack is not a signal-synthesis attack but a delay attack. This type of attack is always possible if only message authentication is used. Hence, this attack would also be possible if a MAC is used. Because of that, the MAC does not improve the security against signal-synthesis attacks, selective-delay or relaying attacks. It only improves the security against counterfeit correction message attacks.

It is possible to authenticate the time message and the station ID without the use of a MAC. The station ID is a fixed value and each transmission time of a signal can be predicted. This makes it possible to embedded the station ID and the transmission time into the key generation of the one-way chain. In the original TESLA, a time interval is given for every key during which the key is valid. It is possible to tighten this time schedule, by defining the exact transmission time of each key. One way to do this, is to exactly define the time interval between two keys and publish the transmission time of the first key of the one-way chain. For example, assume that the first key was revealed at 00:00:00 and every second one key is revealed. If you receive the key k_{73}, you know that this key was sent 73 seconds - 1 second =72 seconds after the first key, so the transmission time was 00:01:12. Note that not every key needs to be revealed by the transmitter. If some of the previous keys have not been published, the current key can still be validated. Therefore, it is

possible to generate a key for every possible transmission time in case that the exact transmission time of each key is not known, e.g. because the data messages vary in size.

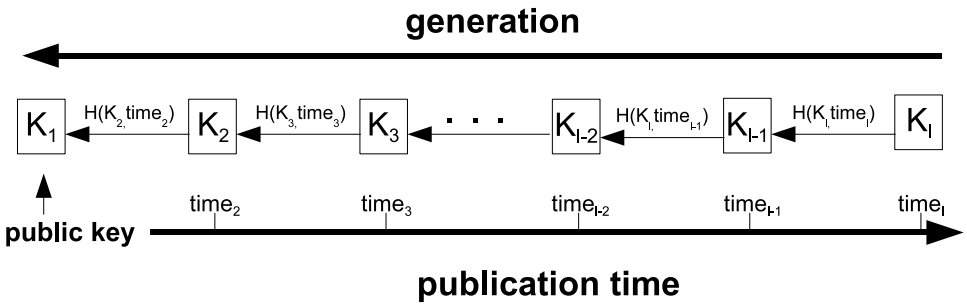


Figure 5.3: The generation and publication of an adjusted TESLA one-way key chain with an embedded timestamp.

To make sure that the time schedule is known to the receiver, it is possible to embed a timestamp into the one-way chain generation. To generate a one-way chain, at first a random number r is chosen, like it is done in the original TESLA. The nodes are generated by adding the timestamp of the transmission time of each node to the input of the one-way function. The first node K_l is computed by using the random number r and the timestamp $time_l$ as an input to the one-way function: $K_l = H(r, time_l)$. The rest of the nodes are generated by computing $K_{i-1} = H(K_i, time_i)$ for $i = l$ to $i = 2$, where $time_i$ denotes the timestamp of the transmission of K_i. The generation and publication of a one-way chain with a timestamp is illustrated in figure 5.3.
To validate a key K_i, the receiver needs to know the transmission time T_i of K_i. As the transmission time is used to compute the position, the receiver already knows the transmission time. The receiver computes the timestamp $time_i$ for the time T_i. With this timestamp, the receiver computes the previous key $K_{i-1} = H(K_i, time_i)$. For $j = i - 1$ to $j = 2$ the receiver computes $K_{j-1} = H(K_j, time_j)$. If K_1 matches the public key, K_i is valid and was sent by the transmitter at the time T_i. As we will see in section 6.1.1, adding a timestamp significantly increases the security of one-way chain.
To make sure that the message comes from the assumed station, each station should have their own one-way chain. This ensures that the receiver knows which station sent out the key.

The big disadvantage of not using a MAC is the missing message authentication. How important message authentication is, strongly depends on the positioning system, the application and the type of messages. Two things need to be considered, when deciding whether or not message authentication is needed.

1. How difficult is a counterfeit correction message attack when using only a one-way chain and no MAC compared to other attacks such as selective-delay or relaying attacks?

2. How dangerous is a counterfeit correction message attack compared to these attacks? Are there attack scenarios that are only possible with counterfeit correction message attacks?

Especially question two is important. In most cases, the same goals that can be achieved with a counterfeit correction message attack can also be achieved with a selective-delay attack. Moreover, selective-delay attacks can spoof a position within the entire coverage area of the received transmitters, while counterfeit correction message attacks can only introduce a much smaller error to the actual position. However, if the LORAN data messages contain additional information that is not directly used by LORAN to determine the position, attacks that are not possible with a selective-delay attack might become feasible. For example, the LORAN data channel can be used to broadcast correction messages for GPS, like it is proposed in [19]. In this case, an attacker could compromise eLoran and GPS at the same time by attacking the LORAN data channel. Therefore, it would be more important to secure the data messages, as an insecure LORAN data channel might compromise the security of other systems.

Sometimes it also depends on the situation whether or not message authentication is needed. For example, if correction messages only correct your position for less than two meters, saving bandwidth by not sending MACs might be reasonable. But if correction messages make a difference of more than 20 meters, securing the correction messages might be beneficial. Another example where message authentication might be needed is when a transmitter station has a malfunction and sends out false navigation data. In this case, the other stations send out warning messages which should be secured, as using the malfunctioned transmitter can cause great position offsets. However, in absent of these abnormalities, it might not be necessary to secure the corrections.
The data channel might not only be used to transmit data for the positioning system, but could also be used to transmit warning messages, for example hurricane or tsunami warnings. For these rare events, message authentication might be necessary as these warnings might have great impacts. Furthermore, forging these messages enable attack scenarios which are not possible with a selective-delay or relaying attack. To be able to switch between message authentication and no message authentication would be the ideal solution. How this can be realized is described in section 6.2.1.

5.4 Using a 2nd channel

Instead of using the LORAN data channel to transmit all authentication messages, another, independent data channel can be used to transmit at least parts of the authentication message. The second data channel can be another radio signal, e.g. transmitted by the DGPS stations. A very promising alternative is to use the Internet as the second channel to publish digital signatures of the LORAN messages. Users can compare these digital signatures with the received LORAN messages. A

counterfeit correction message attack would become impossible, because the receiver can detect every change of the data messages. However, publishing digital signatures of the data messages is not enough to prevent signal-synthesis attacks. An attacker can try to predict the LORAN data messages and transmit these predictions. If the attacker's prediction is correct, the digital signature published in the Internet would match with the attacker's messages. Because the most important information in a positioning system is the arrival time of a signal and not the content of the data message, this would enable a signal-synthesis attack. Whether an attacker succeeds, depends on the probability that an attacker can predict the data messages. Time or almanac messages are very easy to predict while correction messages might be a bit harder to predict. However, over a short time period, these correction messages might not change. This can make it easy to predict the correction messages.
These problems can be avoided by introducing randomness into the data messages. If the data message includes a (pseudo) random sequence, an attacker will not be able to predict this sequence and will not be able to send out valid messages. This randomness can be generated by a random-number generator. A one-way chain or the transmission of a part of the digital signature over LORAN and the other part over the second channel would also provide randomness. One solution is to transmit the keys of the one-way chain over eLoran and a digital signature over another channel, e.g. the Internet. Receivers could verify the one-way chain keys without using the second channel. To authenticate the data messages, the second channel would be needed. This solution has the big advantage that the bandwidth for the MAC or digital signature on the LORAN data channel can be saved. Every user would be able to detect signal-synthesis attacks, while user with access to the second data channel would also gain full message authentication.

5.5 Choosing an authentication method for eLoran

Three criteria are used to make a choice between the candidates. First of all, the security of the underlying system should be well known. The second criteria is the bandwidth needed for authentication purpose and the resulting authentication delay. The third criteria are the performance and hardware requirements.
The security of DSA, ECDSA and TESLA have all been well studied and no security gab that speaks against the use of one of these methods is known today. The big disadvantage of adjusted TESLA is that it needs synchronized clocks as a security requirement. ECDSA and DSA do not rely on synchronized clocks. However, if the clocks are not synchronized, attacks such as selective-delay or relaying attacks are very easy, regardless of the used message authentication method. Therefore, loosely synchronized clocks are a reasonable requirement.
The big advantage of TESLA is the small signature size. An adjusted TESLA signature with a security equivalent to an ECDSA or DSA signature of 320 bits has only about 120 bits (see table 5.1), making adjusted TESLA the most efficient signature scheme of the candidates. ECDSA and DSA seem to have the advantage that the messages can be verified without any delay. However, in practice the authentication time is much shorter using adjusted TESLA. This is due to the fact

that transmitting a ECDSA or DSA signature takes significantly more time, due to the longer signature. This time is much bigger than the authentication delay used in adjusted TESLA. As the transmission of a 41 bit payload message can take up to 2.4 seconds, transmitting 320 bits takes 19.2 seconds if 100% of the bandwidth is available for authentication purpose. With an authentication bandwidth of 7.5%, it takes $(2.4s/7.5\%) \cdot 8 = 256s$ to transmit all parts of the 320 bit DSA or ECDSA signature. The authentication delay of TESLA and a bandwidth of 7.5% is only 96 seconds. (Using a 80 bit key and a 40 bit MAC, see section 6.2) With a bandwidth of 100% it only takes 7.2 seconds to transmit the 120 bits. (But in this case the security delay is only 2.4 seconds) If only a one-way key chain is used without a MAC, the authentication time is 4.8 seconds in the case of 100% available bandidth and 64 seconds if 7.5% of the bandwidth is availiable for authentication purpose.
Even an authentication delay of 19.2 seconds is very long and might be too long for some applications. For other applications an authentication delay of 96 seconds might be still acceptable. However, an authentication delay of 256 seconds is not acceptable for most applications. The significantly smaller authentication time is the main reason for choosing adjusted TESLA.
As adjusted TESLA only uses symmetric algorithms, the hardware requirements and the computation time is better compared to DSA or ECDSA. The only drawback is the case that the receiver had no valid signal from a particular station for a very long time (years). In this case, the computation time is much higher. (see section 6.1.1)

Year	symmetric keysize	adjusted Tesla signature size (key size)	ECDSA signature size (key size)
2020	86	115 (75)	322 (161)
2035	97	125(85)	368 (184)
2045	105	131 (91)	396 (198)
2060	116	145(105)	442 (221)
2075	128	155 (115)	486 (243)

Table 5.1: Comparison of the signature sizes that are expected to be secure in each year according to Lenstra's model. [15] The values for adjusted TESLA are discussed in section 6.1.1. $s = 1998$ is chosen for adjusted TESLA. For ECDSA Lenstra's default value of s=1882 is used. The MAC size in adjusted TESLA is chosen to be 40 bits. The signature size of ECDSA is twice the key length. For more details why the key size of adjusted TESLA can be smaller than that of a symmetric cipher, please see section 6.1.1.

Because of the very small available data bandwidth for authentication purposes, adjusted TESLA is the most promising authentication mechanism for eLoran. However, if the data bandwidth for authentication messages of eLoran can be significantly increased, for example by adding a second data channel, ECDSA is a very promising alternative, as it does not need synchronized clocks.

6 Implementing adjusted TESLA in eLoran

In this chapter I will present my design and security analysis of adjusted TESLA. The goal of adjusted TESLA is to authenticate the transmission time and source of the LORAN signals. In this way, selective-delay attacks can be prevented, because an attacker will not be able to generate signals himself. To protect against counterfeit correction message attacks, the data messages need to be authenticated as well. This can be achieved with adjusted TESLA, in case a MAC is used.

6.1 Choosing the algorithms and bit sizes for adjusted TESLA

6.1.1 For the one-way key chain

Key distribution is a big problem in eLoran. To distribute a new public key, either a second channel is needed, or the new public key must be sent using the LORAN data channel. If the LORAN data channel is used, the key must be frequently sent out, as it might be possible that receivers have no connection to the data channel for a long time. These keys also need authentication, so that no attacker can create their own public keys. Therefore, a digital signature would be needed to authenticate the public keys, which would increase the size of the transmission. This would increase the needed bandwidth for authentication purposes a lot. Hence, using only one key which will be valid for many years is the better option. However, this results in the requirement that the public key needs to be secure for a long time and that nodes of the one-way chain can be computed very efficiently.

Attack models

This section discusses how an attacker can break the public key of a one-way chain. There are two basic attack models:

1. The attacker breaks the one-way property using a weakness in the used one-way function.

2. The attacker tries to guess a node of the one-way chain.

Attack model 1: Breaking the one-wayness
Assume the one-way chain uses the one-way function $H : \{0,1\}^s \rightarrow \{0,1\}^s$ with $k_i = H(k_{i+1})$, and k_1 being the current public key. If a secure one-way function

is used, the attacker has no other option to find a k_{i+1} for a given k_i than guessing k_{i+1}. If each k_i occurs with the same probability, the chance that an attacker guesses correct is $1/2^s$, so on average $2^s/2$ operations are needed. However, the used one-way function might not be completely secure. As described above, the used key needs to be secure for a long time. So even if a one-way function is chosen for which no attack is known today, a weakness in this function might be exposed in the future. Therefore, it might be possible to compute a k_{i+1} for a given k_i in $u < 2^s/2$ operations. But computing a key k_{i+1} does not automatically mean that the system is broken. Each key will only be valid for one second. If it takes the attacker one second or more to compute $k_2 = H^{-1}(k_1)$, with $H^{-1}()$ denoting the inverse operation of the one-way function, the attacker has not gained any secret information, as k_2 will already be considered old. To be able to compute a key that is still valid, the attacker needs to be able to break the one-way function in less than a second. With a sufficient length of s, this will only be possible if a very big weakness is discovered in the used one-way function. If a well discussed one-way function, which has not shown any weakness yet, is used, this is very unlikely to happen. Hence, the biggest threat is attack model 2, the guessing-attack.

Attack model 2: Guessing-attack
In this attack, the attacker tries to guess a key k' with $H^i(k') = k_1$ for some i, where k_1 is the current public key. The current public key is the last authenticated key by the receiver. If the output of the one-way function H is random, an attacker can compute $k'_{i+1} = H(k'_i)$ until either $k'_{i+1} = k_1$ or $k'_{i+1} = k'_j$ for some $j \leq i$. In the case of $k'_{i+1} = k'_j$ for some $j \leq i$, the one-way function generated a circle, and the attacker needs to choose a new random number k'' as his seed guess. The attacker will need 2^s operations on average to find a key. Just like it is in the attack that brakes the one-way property, the attack will be useless if the attacker reveals a key which has already been published since he started the attack. However, in this attack the attacker does not just hash random numbers and compares them with k_1, but computes one-way chains himself. The chance that the one-way chain will enter a circle, is the same as the chance that a collision in a hash function occurs. If the one-way function can be seen as a random function, on average a collision will occur in $2^{s/2}$ operations, due to the birthday paradox. The chances that i is bigger than the length l of the one-way chain is very high, as $l << 2^{s/2}$. In this case, the attacker would have broken the entire system, because the attacker would possess a one-way chain which is even longer than the original one-way chain.

To make this attack more difficult, a counter should be inserted into the one-way chain. If the one-way function $H : (\{0,1\}^s, \{0,1\}^m) \rightarrow \{0,1\}^s$ also includes a counter $i \in \mathbb{N}$, so that $k_i = H(k_{i+1}, i)$, an attacker can not simply apply the one-way function on the last computed value. If he wants to find a value k'_i with $k'_j = H(k'_{j+1}, j)$ for $1 \leq j \leq i$ and $k'_1 = k_1$, he needs to randomly pick a k'_i and compute k'_1 for one previously chosen i. If k'_1 does not match k_1, he needs to guess another k'_i. To compute k'_1 he needs i operations. Hence, on average the attacker will need $i \cdot 2^s/2$ operations. This is an increase in computation complexity in the order of i. It makes

it much more unlikely that an attacker will find a k'_i with a big i. As the lifetime of the revealed key depends on i, a key with a small i will not last very long. If the used one-way chain has the length l, in the scenario without a counter, a successful attacker will, with a very high probability, receive a one-way chain of a length bigger than l. Thus he will have broken the entire system and an attacker would only need to break the system once. But if a counter is used, the attacker will receive a one-way chain of the length i. As the complexity of the attack increases linear with the size of i, an attacker will only be able to break a one-way chain using a small i. Therefore, the attacker will only possess i keys, so that an attacker would only be able to break the system for a very limited time period. In this implementation, there is one key for every second, regardless whether or not a key will be published in this second. Therefore, a one-way chain of the size i will only last i seconds.

If it takes an attacker more than one second to compute 2^s operations, he will need $t > i$ seconds to compute $i \cdot 2^s/2$ operations and therefore calculate a key which is already expired. Hence, if the attacker can not compute $2^s/2$ operations per second, the attacker needs to frequently update the public key k_1 which he wants to break with the last published value. This makes the attack non-deterministic and more complex as the search key needs to be updated frequently. On average $2^s/2$ operations are needed to find a s bit key. However, if after each try the search key is changed, on average 2^s operations are needed. Hence, on average $i \cdot 2^s$ operations are needed to find a valid key of the one-way chain because the public key needs to be updated frequently if a counter is used.
Although, there will be one key for every second, not every key will be revealed. Depending on the message schedule, a key will probably only be revealed about every 60 seconds or more. Therefore, an attacker can not update the public key every second and needs to choose a larger i.
All this shows that adding a counter into the one-way chain generation process increases the security of the system significantly. A timestamp, as it is suggested in section 5.3, can be used as such a counter.

Key size

For breaking the one-wayness (attack model 1), an attacker needs to be able to compute $2^s/2$ operations within one second if the used one-way function is secure. In the guessing-attack (attack model 2), an attacker needs $i \cdot 2^s$ operations if a counter is used, but this attack can take longer than one second and will still be successful. To predict how long exactly a key length of s bits will be sufficient can only be predicted based on assumptions gained from the past. Symmetric ciphers below 60 bits can already be broken using equipment that costs less than 12,000$. [7] It is believed to be computational infeasible to break an 80 bit key at the moment. [3] [15]
The big question is which key length will be secure in the future for the generation of a one-way chain. To answer this question, the model provided by Lenstra in [16] is used. The model is based on the widely accepted Moore's law. According

to Moore's Law, the density of components per integrated circuit doubles every 18 months. A widely accepted interpretation of this law is that the computing power per chip doubles every 18 month.[16] Moore's law is not based on any physical law but only on the observation of the past developments. However, it has proved to be correct for 40 years now and it is believed to be a good assumption for the future.

Lenstra's model:[16]
To make the model adjustable, Lenstra's model depends on five parameters. (for asymmetric key length there are three additional parameters which we do not need)

`Parameter` s`:` The security margin s is defined as the year until which a user was willing to trust the DES.
`Parameter` m`:` The variable $m > 0$ is defined as the number of months it takes on average for an expected two-fold processor speed-up and memory size increase.
`Parameter` t`:` The (0,1)-valued variable t defines how m must be interpreted:
- If t = 1, the amount of computing power and random access memory(RAM) one gets for a dollar is expected to double every m months.
- If t = 0, the amount of computing power and RAM is expected to double every m months, irrespective of the price.
`Parameter` b`:` The variable $b > 0$ is defined as the number of years it takes on average for an expected two-fold increase of budget. The default value for b is 10 years.
`Parameter` v`:` The variable $v > 0$ is defined as the ratio of the number of cycles required for a single block encryption using the DES and the symmetric key system the user wishes to use.

The equivalent symmetric key length *length* in bits for a year y is computed with the formula

$$\begin{aligned} length &= 56 + log_2(5 \cdot 10^5 \cdot 2^{12(y-s)/m} \cdot 2^{t(y-s)/b}/(5 \cdot 10^5 \cdot v)) \\ &= 56 + (y-s)(12/m + t/b) - log_2(v) \end{aligned}$$

To make the model more adjustable, Lenstra defines m as the number of month it takes on average for an expected two-fold processor speed-up and memory size increase. According to Moore's law, m equals 18. As this value has proven to work best over the last 40 years, in this work we have chosen m according to Moore's law as $m = 18$.
We will assume that one computation of our one-way function will roughly take the same time as one DES encryption and therefore choose $s = 1$.

In Lenstra's model the variable s defines the wanted security level. This is the most important value, because it gives us concrete results on how secure our one-way chain will be in the future. For example, if we choose $s = 1982$ the resulting key length will be as secure as the DES (56 bit key length) was in 1982. To choose a good value for s is crucial to get a meaningful result. Therefore, a detailed analysis is provided of the security of a one-way chain generated with DES for each year in

the next section.
The reason why Lenstra chose this parameter as a security margin is that the security and history of DES is quite well known. As a default value for s, Lenstra chose $s = 1982$. In 1980 a hardware attack on DES was assumed to require \$50 million and would take two days. [16] Hence, DES was brakeable and some security agencies might have already broken DES, but for commercial applications DES was still considered secure enough. In 1993, Wiener proposed a hardware architecture for 1 million dollar that could brake DES in 3.5 hours(but which was never build). In 1998 DeepCrack was build with a budget of 250.000\$ that can break DES within 112 hours.[23] In 2006 the COPACOBANA project developed a FPGA-based DES cracker that needs on average 6.4 days to break DES and only used commercially available hardware for no more than 12,000\$. [14][7]
To get an impression how secure a one-way chain based on DES would have been for these years, we will estimate the complexity of an attack using the described DES crackers. To be able to perform attack model 1 (breaking the one-wayness), an attacker would need to break DES in less than a second. In 2006, an attacker could break DES in 6.4 days with 12,000\$ hardware. Therefore, an attacker would need about $12{,}000\$ \cdot 6.4 \cdot 24 \cdot 60 \cdot 60 = 6,635,520,000\$$ to build a DES cracker that needs only one second to break a DES key. This makes breaking the one-wayness impracticable for 2006.
To estimate the time and hardware requirements for the guessing-attack, an appropriate value for i needs to be chosen. If we choose $i = 1$, the attacker has one second to mount the attack after a correct key was found, minus the time it took to compute the key. This scenario is impossible, as the receiver needs to synchronize with the spoofed signal first. The attacker would also not be able to create a valid MAC, as the attacker needs to know the key d seconds ahead of time to create a valid MAC, where d is the minimum delay between the MAC and the corresponding key. If the authentication bandwidth is 7.5% and the next key will be the authentication key, d is 32s. However, this means that the attack is mounted immediatly after a key is found. In practice, the attacker needs to transmit the key to the spoofing device and the spoofing device needs to take control over the LORAN data channel before this attack can be started. We assume that the attacker needs at least about 30 seconds to start the attack. Therefore, we assume that i needs to be at least 60. However, another factor that increases the complexity of the attack is the fact that an attacker will not be able to update the search key every second. Although there will be a key for every second, only about every minute a key will be released. Therefore, the attacker will only be able to update a key every 60 seconds or more depending on the key schedule. (see section 6.2) To be able to compute keys that last for at least 60 seconds an attacker needs to choose $i = 60s + 60s = 120s$.
As the complexity increases linear with i, an attack with $i = 120$ would take $6.4\text{days} \cdot 120 = 768$ days using the COPACOBANA hardware for 12,000\$. During all the time the attacker needs to be prepared to start the attack within 30 seconds. The attacker would also only be able to forge one MAC-Key pair from one transmitter station. For nearly all applications such an attack is impossible. A reasonable time the key search can take if the attacker has 30 seconds to start

the attack after the key is found would be at least below one day. (Note that the attacker would only be able to spoof one MAC-key pair) In this case, a DES cracker capable of this attack would have cost $12,000\$ \cdot 768 = 9,216,000\$$ in 2006 if the attack takes one day. This assumes that an attacker uses special purpose hardware to break DES, which is the most cost-effective way to break DES (and AES). However, in recent years several criminal organizations managed to get control over thousands of computers in so called bot-networks using maleware. This can provide these organization with computation power of thousands of computers. In 2006, about 30.000 Pentium 4/3GHz PCs were needed to compute a DES key in the same time as COPACOBANA. Hence, $30,000\text{PCs} \cdot 768 = 23,040,000\text{PCs}$ would be needed to break DES within one day with $i = 120$ in 2006. It should be noted that using a bot-network for this attack is logistically difficult, as every 60 seconds these PCs would need to be updated with the new public key.
Note that the attacker will only gain keys valid for one transmitter in this attack. But for a signal synthesis attack, the attacker needs the keys of at least three transmitter stations to generate the needed three navigation signals. As the attacker will only find keys for one station that are valid for 120 seconds, it is very unlikely that the attacker finds three keys that are valid for the same 120 seconds if one attack takes about one day. Hence, if the attacker only has the keys for one transmitter station, the attacker cannot use this information for a signal-synthesis attack, but only for a counterfeit correction message attack. However, depending on the transmitted information that could be enough to attack the system. But it is more likely that the attacker does not profit much if he has only valid keys for one transmitter station.

Instead of using a small i and a short attack time of one day, it might be reasonable that an attacker accepts an attack time of several days if the attacker receives keys that last longer. Assume that the attacker wants keys that last 1 day, then i needs to be $i = 60 \cdot 60 \cdot 24 = 86,400$. With a budget of one million dollar it took about 18 years to break 86,400 keys in 2006. [1] If the attack time is reduced to one year, 18 million dollars would be needed. Table 6.1 and figure 6.1 give an overview of different parameters for attacks on one-way chains based on DES in the year 2006.

These results show that although it was rather easy in 2006 to break DES, breaking an adjusted TESLA one-way chain based on DES was still very difficult. A lot of money and skills would have been needed to successfully attack one signal for a short time. Attacking more than one one-way chain for the same period of time would have been extremely difficult, as several billions of dollars would have been needed. Hence, for most scenarios an attack would have been very unrealistic in 2006. Note that the values provided above are just estimations. COPACOBANA was build using re-programmable commercial FPGAs. With a budget of 1,000,000$ it probably was possible to build an asic based DES cracker which is more efficient

[1] For one million dollar an attacker could get 83 COPACOBANAs in 2006. It takes 6.4 days to break one key with one COPACOBANA so that it takes 6.4/83 days to break one key with 83 COPACOBANAs. Because 86,400 keys are needed, the attack takes $6.4\text{days}/83 \cdot 86,400 \approx 6662\text{days} \approx 18\text{years}$

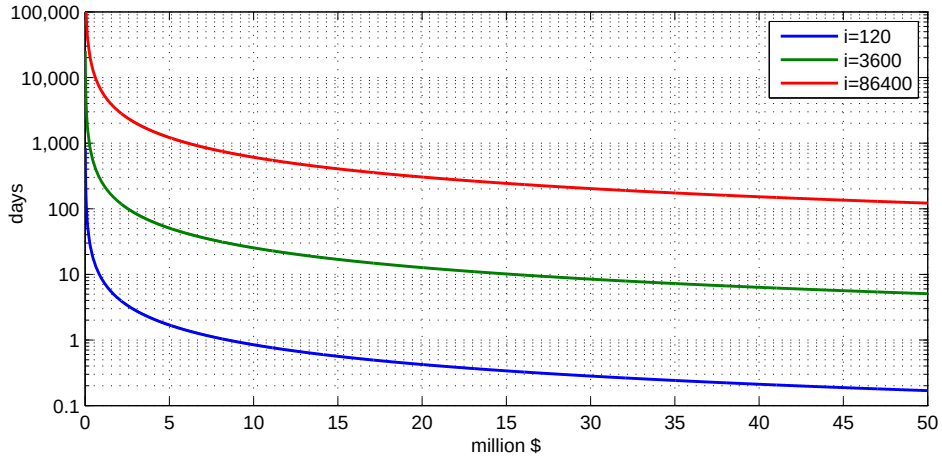


Figure 6.1: The average attack time in days to find i keys with a key size of 56 bit for $i = 120$, $i = 3600$ and $i = 86400$ in 2006.

in breaking DES than 83 COPACOBANAs. Table 6.2 and figure 6.2 show the same analysis for the year 1998 and DeepCrack as a reference.

$	i=120	i=3,600	i=86,400
50,000	184.32 days	5529.6 days	132710.4 days
100,000	92.16 days	2764.8 days	66355.2 days
500,000	18.43 days	552.96 days	13271.04 days
1,000,000	9.22 days	276.48 days	6635.52 days
5,000,000	1.84 days	55.3 days	1327.1 days
10,000,000	0.92 days	27.65 days	663.55 days
50,000,000	0.18 days	5.53 days	132.71 days

Table 6.1: Attack time according to the attacker's budget for an attack revealing i keys of a one-way chain based on DES using hardware from 2006.

Three settings are used to compute the needed key-length. In the first setting Lenstra's default value $s = 1982$ is used. In the second setting $s = 2006$ and in the third $s = 1998 ischosen$. Table 6.3 shows the results of this analysis.

Of course, normally you should always try to have a security margin and if the security is questionable move on to a longer key. However, as eLoran has such a small data rate, a fair balance between security and system parameters such as authentication time and bandwidth is needed. It is useless to implement a security mechanism which is perfectly secure but most applications won't use the mechanism as the time to authenticate is too long. Because one eLoran message has 41 bits payload, the key sizes should be a multiple of 40 (one bit can be used to identify the message). Therefore, an 80 bit keysize or a 120 bit key size are the best options. In 2037, an 80 bit key will provide the same security as a 56 bit key in 2006. Although breakable, such an attack would require a lot of skills and resources, with the result

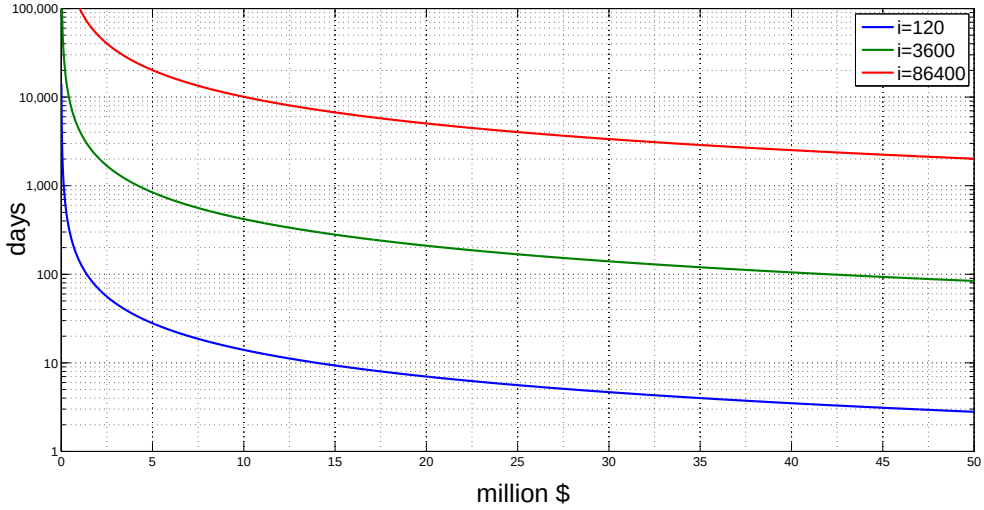


Figure 6.2: The average attack time in days to find i keys with a key size of 56 bit for $i = 120$, $i = 3600$ and $i = 86400$ in 1998.

that this attack is impracticable in almost all scenarios. It should be noted again that an attacker would only be able to break one chain at a time. Breaking more than one chain at a time will be still computational infeasible even if attacking one chain is possible. An attacker would need to be able to break a 80 bit or 120 bit key in about one second to be able to attack more than one chain at the same time. However, theoretically an 80 bit key for one chain could be attackable within the next 20 years, although huge amounts of resources would be necessary. If a 120 bit key is chosen, according to Lenstra's default setting, a 120 bit key is secure until 2065. The attacks described for a 56 bit keylength in 2006 would probably not be feasible until 2089. Hence, for higher long term security a 120 bit key needs to be chosen. However, an 80 bit key will also be reasonably secure for a long time for most applications. Instead of choosing an 80 bit key size it is also possible to use an 88 bit key instead and reduce the MAC size from 40 bit to 32. An analysis of the security of a 40 bit and 32 bit MAC is presented in section 6.1.2.

Algorithm

To create a secure one-way chain, a secure one-way function which will last very long is needed. Of course it is not possible to predict which one-way function might be broken in the future. The best thing one can do is to choose a one-way function which is very unlikely to be broken. As there are no measurements on the security of a one-way function, the only option is to choose a one-way function, which is well studied and has not revealed any weakness so far. There are two popular ways to

$	i=120	i=3,600	i=86,400
50,000	2800 days	84000 days	2016000 days
100,000	1400 days	42000 days	1008000 days
500,000	280 days	8400days	201600 days
1,000,000	140 days	4200 days	100800 days
5,000,000	28 days	840 days	20160 days
10,000,000	14 days	420 days	10080 days
50,000,000	2.8 dayss	84 days	2016 days

Table 6.2: Attack time according to the attacker's budget for an attack revealing i keys of a one-way chain based on DES using hardware from 1998.

key length	s=1982	s=1998	s=2006
75	2006	2022	2030
80	2013	2029	2037
85	2019	2035	2043
88	2023	2039	2047
90	2026	2042	2050
100	2039	2055	2063
110	2052	2068	2076
120	2065	2081	2089
128	2075	2091	2099
130	2078	2094	2102
140	2091	2107	2115
150	2104	2120	2128
160	2117	2133	2141

Table 6.3: The year and key length which provide equivalent security as 56 bit key provides in the year s.

generate a one-way chain. Either a hash function or a block cipher is used. The most popular hash functions today are the SHA family, SHA-1,SHA-224, SHA-256, SHA-384 and SHA-512. The best known and widely used block cipher today is AES. So either a hash function from the SHA family or a one-way function based on AES is most promising.
Recently, researches have shown that it is possible to break the collision resistance of SHA-1.[31] Collision resistance means that it is impossible to find two inputs x_1, x_2 with the same hash value $H(x_1) = H(x_2)$. This property is not needed to create a one-way chain. Only preimage resistance is needed, which means that it is impossible to find the input x for an output y with $y = H(x)$. If a hash function is used to generate the one-way chain, not the entire output of the hash function is used but only 80 or 120 bits. Therefore, another security property, partial-preimage resistance, is needed. In [20] partial-preimage resistance is defined as followed:

> It should be as difficult to recover any substring as to recover the entire input. Moreover, even if part of the input is known, it should be difficult

to find the remainder (e.g., if t input bits remain unknown, it should take on average 2^{t-1} hash operations to find these bits.)

There are no attacks against preimage resistance or partial-preimage resistance against the SHA family known. But nevertheless, the attack on the collision resistance can be seen as a warning that SHA-1 does not behave like a random pattern. So far, no attack against any security property of AES is known. This does not mean that breaking the preimage-resistance of SHA-1 is easier than breaking AES. But because of the lack of other measurements, the best option is to go with an algorithm which has not failed any of its security goals.
Besides the security aspects, the one-way function needs to be very fast and should not require much resources. A different one-way chain will be used for every station. But a receiver will not be able to receive every station all the time. Therefore, the receiver might not have received a fresh key of the one-way chain from a station for a long time. Let the last key the receiver has authenticated be k_a. If the receiver needs to authenticate a key k_{a+t} which was published t seconds after k_a, the receiver needs to compute t operations of the one-way function. The time t will be very long if either a receiver was offline for a long time, or if the receiver's position is moved to an area it has not been for a long time. If the receiver was offline for a long time, it is reasonable to give the receiver some time to generate the current key. More problematic is the other case, in which the receiver moves to a position where it has not been for a long time. In this case, the time to compute the key and therefore the time needed to authenticate should be as small as possible. To get an idea of the number of operations needed, assume that the newest valid key the receiver possesses for the needed one-way chain is two years old. For example, such a time is quite reasonable for ships or airplanes that were used in a different area for the last two years. In this case, $t = 60 \cdot 60 \cdot 24 \cdot 365 \cdot 2 \approx 2^{25}$ operations need to be computed by the receiver. (As one key is used every second) In most cases this computation will be done in hardware, but there might also be situations where software receivers are used. Hence, a very good hardware performance is needed, while software performance should not be too bad as well.
Hash functions from the SHA family are designed to hash long data messages in a short time. However, they were not designed to hash messages of an input size below 512 bit efficiently. SHA-1 uses pattern to extend the input to 512 bit and uses 80 rounds, even if the input size is 80 or 128 bits. On the other hand, AES is designed for input blocks of only 128 bits and uses only 10 rounds. AES is also designed to be very efficient, both in hardware and software. Therefore, a one-way chain based on AES is much faster.
Because of the better performance and the well discussed security properties, the best choice for the one-way chain is a one-way function based on AES. We will now discuss the different ways to generate the one-way chain using AES. Each node of the one-way chain should be either 80 or 120 bits, depending on the chosen security level. The input to each node should be the previous key k_{i+1} and the timestamp t_i of 40 bits. We define the one-way function as $k_i = f(k_{i+1}, t_i)$.

`Method 1: fixed-key`
This method is only usable with a key size of 80 or 88 bits. The reason why it can not be used for 120 bit keys is discussed at the end of this paragraph. The algorithm is as followed:
The output of function $g : \{0,1\}^{128} \rightarrow \{0,1\}^{80}$ is defined as the 80 most significant bits of the input. For every station, choose a unique public 128 bit string S. To generate a one-way chain with n elements, choose an 80 bit random number k_l as the seed value. For $l \leq i \leq 1$ compute $k_{i-1} = g(enc_S(k_i||00000000||t_i)) \oplus k_i$, where enc_S denotes the 128-Bit AES encryption with the key S, $||$ denotes the concatenation of the bit-strings and $\oplus$ denotes the bit-wise exclusive-or operation. Figure 6.3 illustrates the generation of the one-way chain.
The security of this construction is proven in [20] under the assumption that the encryption $y = enc_k(x)$ is a random permutation:

> For any choice of y, finding any x (and key k) such that $enc_k(x) \oplus x = y$ is difficult because for any chosen x, $enc_k(x)$ will be essentially random (for any key k) and thus so will $enc_k(x) \oplus x$; Hence, this will equal y with no better than random chance. By similar reasoning, if one attempts to use decryption and chooses an x, the probability that $enc_k^{-1}(x \oplus y) = x$ is no better than random chance. Thus $f(x)$ appears to be a one-way function.

Note that AES does not behave like a random function, because otherwise encryption and decryption would be equivalent to looking up values in a large table of random numbers. Block ciphers such as AES are bijections, and thus at best exhibit a behavior more like random permutations than random functions.[20]
The big advantage of this construction is that the same AES-key is used for every node during the generation of the one-way chain. Therefore, the key schedule only needs to be generated once, and can be reused for every node. This enables a more efficient implementation of the one-way chain computation. If a 120 bit key is used, this method can not be used, as the concatenation of a 120 bit key k_i and the 40 bit timestamp t_i results in a 140 bit String, which is longer than the allowed 128-bit block size of AES. One might think to use exclusive-or instead of concatenation to link the keys and the timestamp could solve the problem. The generation of the one-way chain would then look like this: $k_{i-1} = g((k_i) \oplus enc_S(k_i \oplus t_i||00000000))$
However, this might result in security problems. For a known key k_i an attacker might be able to generate a pair of k', t' with $k_i = g((k') \oplus enc_S(k' \oplus t'||00000000))$. To do that, the attacker picks a 120 bit value a. Then the attacker encrypts this value resulting in $enc_S(a||00000000)$. The attacker now computes $k' = enc_S(a||00000000) \oplus k_i$. After that, the attacker can compute $t' = k' \oplus a$. Because the timestamps t_i are only 40 bit values, a and k' need to be equal in the 80 least significant bits (If the 40 most significant bits are used for the $k_i \oplus t_i$ operation). The probability that these bits are equal is $1/2^{80}$. So an attacker will only need on average 2^{80} tries to find a pair of k' and t'. Note that an attacker can do the encryption of a offline and that he can reuse these values for every attack. To make an attack more difficult, instead of using $k_i \oplus t_i||00000000$ as the input to AES, $t_i \oplus (00000000||k_i)$ can be used as

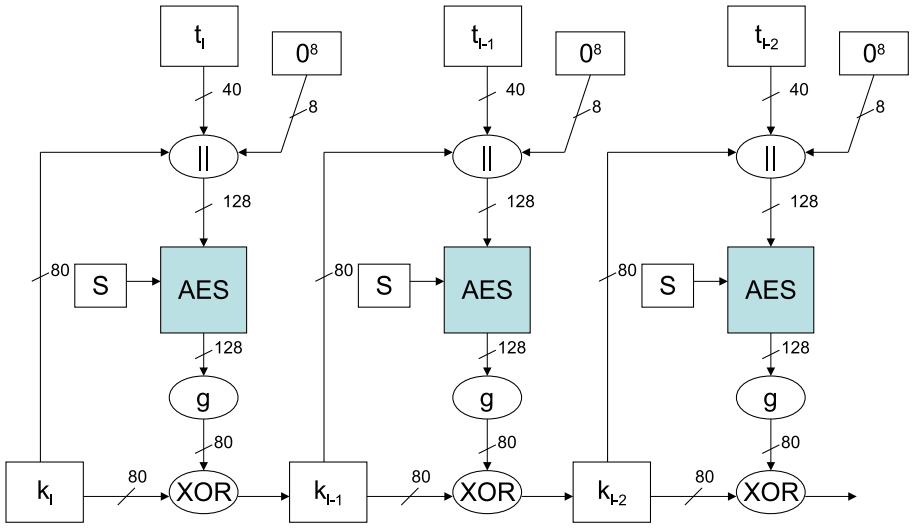


Figure 6.3: The construction of the one-way chain using the fixed-key method (method 1).

an input to AES. In this way, an attacker would have a chance of $1/2^{88}$ to guess a correct k' instead of $1/2^{80}$.
If an attacker can compute the two values k' and t', it does not automatically imply that these values are of any use to the attacker. If the timestamp used for computing k_i is t_i, then a verifier can check if t' equals t_{i-1}. The chance that these two values are equal are $1/2^{40}$ (or $1/2^{32}$ if no padding is used) so that the overall complexity of the attack would still be O(120). Nevertheless, the verifier might be weakly implemented, so that the verifier only compares the timestamp with the current time. In this case, an attacker could jam the receiver, until the time of the forged timestamp is equal to the current time and send the forged *key*. If the attacker only checks if the resulting key-chain results in the last verified public key, this attack might be possible. This scenario is very unlikely to be practical but there might be other ways how an attacker can use this weakness in the construction of the one-way function. To ensure security it is best to choose a construction which does not have any known weakness. However, the advantage of this construction would be the speedup during the computation of the one-way chains, as the same key is used for every encryption.

`Method 2: timestamp-as-the-key`
The main idea is to use the timestamps as the AES-key to link the nodes k_i to the timestamps t_i. The rest of the construction is the same as in the fixed-key method: The output of function $G : \{0,1\}^{128} \rightarrow \{0,1\}^{120}$ is defined as the 120 most significant bits of the input. To generate a one-way chain with l elements, choose a 120 bit random number k_l as a seed. For $l \leq i \leq 1$ compute $k_{i-1} = g(enc_{t_i}(k_i||00000000)) \oplus k_i$, where enc_{t_i} denotes the 128-Bit AES encryption with the key t_i, $||$ denotes the concatenation of the bit-strings and $\oplus$ denotes the bit-wise exclusive-or operation. This method is illustrated in figure 6.4.
The idea behind the security of this method is the same as the idea behind the fixed-

key method. The security of this construction does not depend on the used AES-key, as the key can be public. The only difference between the fixed-key method and timestamp-as-the-key method is the use of the timestamp as the AES-key instead of a fixed string and the used key length. Hence, this construction is secure under the assumption that the AES encryption behaves like a random pattern.

The disadvantage of this construction is that for each AES-encryption another key

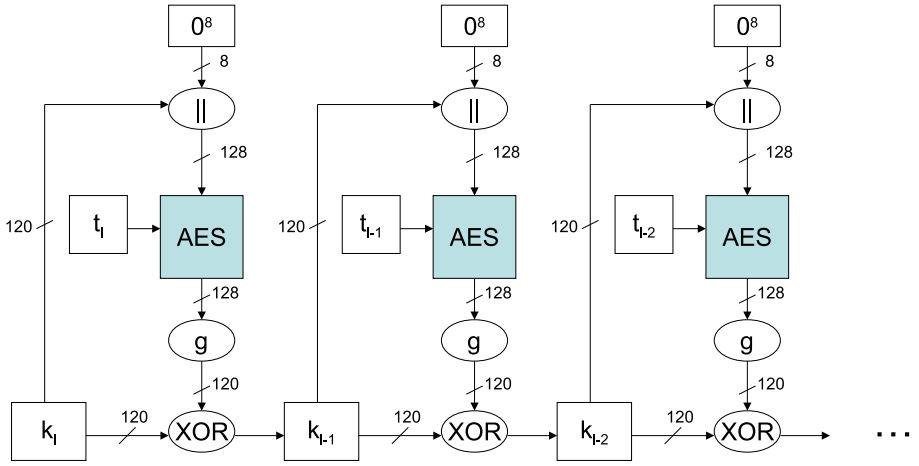


Figure 6.4: The construction of the one-way chain using method 2, the timestamp-as-the-key method.

is used. Each time a new key-schedule is needed which prevents more efficient implementations of AES. If a key-size of 80 or 88 bits is used, the fixed-key method is the preferred choice. Because of the weakness of the fixed-key method in constructing 120 bit key chains, the timestamp-as-the-key method is used for a key length of 120 bits.

6.1.2 For the MAC

MAC length

This section focuses on the needed length for the MAC. At first, the possible attack scenarios on the MAC are described. With this information the complexity of attacks on the MAC can be examined and the needed MAC length can be determined. Assume the valid MAC for the message m and the current, not yet published key k is $mac = MAC_k(m)$. Furthermore, assume that the MAC will be sent at time t_m, and the key k will be revealed later at time t_k. As a requirement of TESLA, the clocks of the receiver and the transmitter are loosely synchronized, so that the receiver will refuse MACs that are generated with the key k after the time t_k.

`Attack 1: key-candidate attack`
In the first attack, the attacker tries to create a valid MAC mac with $mac = MAC_k(m') \neq MAC_k(m)$ for some message m' by revealing the secret key k. To break the secret key k, the attacker can either use a weakness in the used MAC to

compute a key candidate k' with $MAC_{k'}(m) = mac$ or he can try to guess key k. A secure MAC should never reveal any information about k if the attacker knows m and mac. So this attack should not be possible if a secure MAC is chosen. The fact that the key is only used once to generate a MAC makes such an attack even harder, as the attacker will not be able to possess more than one k mac pair. If the attacker can not use a weakness in the MAC algorithm, the attacker needs to guess k. The attacker can check his guessed key k' by computing $mac' = MAC_{k'}(m)$ and comparing mac' with mac. The chance that $mac' = mac$ is $1/2^r$, with r being the MAC length in bits because there are 2^r possible outputs of the MAC algorithm. If $mac' = mac$, k' is called a key candidate. The attacker can check if the key candidate k' is valid by verifying the key candidate using the public key of the one-way chain. The probability that the key candidate k' is the actual key k depends on the key size and the MAC size. Let the key length be s bits and the MAC length be $r < s$ bits. The chance that $k' = k$ if $MAC_{k'}(m) = MAC_k(m)$ is $1/(2^s/2^r) = 1/2^{s-r}$. Hence, if $s = 80$ and $r = 40$ the chance is $1/2^{40}$ that the guessed key is correct. If $s = 120$ and $r = 40$, the chance that the guessed key is correct is $1/2^{80}$. If the attacker can only guess key candidates, the overall chance to guess the correct key stays $1/2^{s-r} \cdot 1/2^r = 1/2^s$. Hence, this attack will be as difficult as attacking the one-way chain. However, if the attacker can use a weakness in the MAC to generate a key candidate k' more efficiently than guessing a key candidate, this attack will be more efficient than a guessing attack on the one-way chain. The smaller the MAC is, the more difficult will this attack become as the chance to get a false key candidate increases.

`Attack 2: MAC-guessing attack`
Besides trying to find the correct key, the attacker can also try to guess a MAC. If each output of the MAC occurs with the same probability, the chance that the attacker guesses correct is $1/2^r$. The problem the attacker faces is that he has no way to test his guess on his own. The attacker does not know the secret key k and therefore can not verify if his guess is correct. The only one who can verify the MAC before the secret key k is published is the base station, as the base station is the only entity that knows the key k. But the base station is a trusted entity and will not validate any MACs. After the key k is released by the base station, the receiver will not accept a MAC with the key k any more. Hence, the attacker needs to send his guess to the receiver, without testing if his guess was correct. A receiver knows that each key will only be used for one MAC, so that the receiver will not accept more than one MAC for each key. In practice, the receiver will also know an upper bound of the number of transmitted MACs during one time period. So the receiver will only verify a limited number of MACs. On average, the attacker needs to guess 2^r MACs until the attacker guesses a correct MAC. If a receiver accepts only one MAC every d seconds, then the attacker needs on average $2^r \cdot d$ seconds until the receiver accepts one of his messages. If 40 bits are chosen as the size of the MAC, there are 2^{40} different outputs of the MAC. If the used MAC is secure, each of these outputs occurs with the same probability. The payload of a eLoran message is 41 Bits. Lets assume that a key length of at least 80 bits is used. In

this case a TESLA message consists of 80bits + 40bits = 120bits. For these 120 bits 3 eLoran messages are needed to transmit the TESLA messages and one additional eLoran message to transmit the forged message m'. Hence, a receiver will have a lower bound of accepted MACs of at least 4 eLoran messages. One eLoran message takes about 2.4 seconds (see section 6.2). This results in a lower bound of transmission time for a MAC of at least $4 \cdot 2.4$ seconds = 9.6 seconds. Therefore it would take at least $2^{40} \cdot 9.6$ seconds $\approx$ 334,706 years on average to guess one MAC.

`Attack 3: message-guessing attack`
Instead of trying to find a MAC mac' that fits to our forged message m', an attacker could also try to find a forged message that results in the same MAC. So the attacker tries to find a message m' with $MAC_k(m') = MAC_k(m)$. The MAC should have the property that the chance that $MAC_k(m') = MAC_k(m)$ should be the same for every $m \neq m'$. If the used MAC fulfills this property, then the chance that the attacker guesses a correct m' is $1/2^r$. This is the same as if the attacker would try to guess the MAC. But to be able to test his guess, the attacker needs the corresponding key k. As this key will not be published until t_k, the attacker does not have any chance to test his guess. Just like in MAC-guessing attack, the attacker needs to transmit his forged message m' without testing if $MAC_k(m')$ matches $MAC_k(m)$.

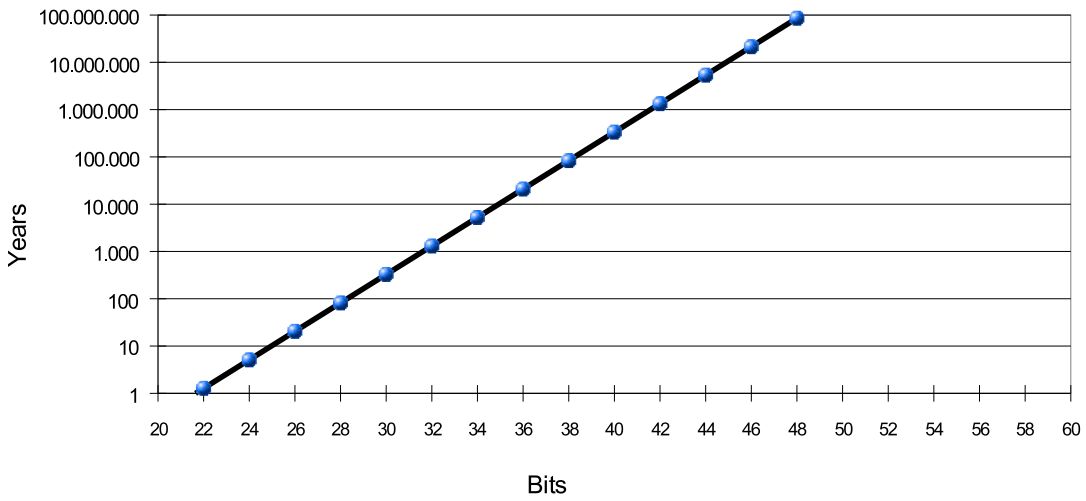


Figure 6.5: The expected time in years it will take to forge a MAC in a message-guessing or MAC-guessing attack in dependence of the MAC size in bits. The receiver accepts one MAC every 9.6 seconds in this analysis.

Hence, the security of the MAC depends on the length of the used key and the length of the MAC. However, the length of the MAC can be quite short, as there is an upper bound of operations per time period the attacker can perform in the MAC-guessing and message-guessing attack. The attacker only knows if the attack was successful after the secret key is revealed. The attacker can not use the information gained during the attack of one receiver to attack other receivers, as the key used during a successful attack will already be expired at the time the attacker

notices his success. Hence, the attacker can not increase the speed of the attack on one particular receiver by attacking many different receivers at the same time. As outlined, this upper bound of operations is at least 9.6 seconds for one operation, although in practice this number will be much higher and will very likely be at least one minute.(see section 6.2) The key-candidate attack even becomes more difficult the smaller the MAC is.
Therefore, a MAC length of 40 bits will provide more than enough security, as it takes at least 334,706 years on average to break one MAC. It is also reasonable to choose a MAC length that is shorter than 40 bits. If a MAC length of 32 bits is used, the average attack time is about 1,307 years. Figure 6.5 shows the expected attack time for the message-guessing attack and the MAC-guessing attack for different MAC lengths.

Algorithm choice

The preferred algorithm for the MAC depends on the used algorithm for the one-way chain. As I propose a one-way chain based on AES, I propose a MAC based on AES, as well. This has the big advantage that the implementation of AES can be reused and there is no need to implement another algorithm like SHA-1. The messages that are signed are very small. Only a couple of hundred bits need to be signed with one MAC. Therefore, the MAC does not need to be very fast. Saving space by reusing the AES implementation is much more efficient than choosing a faster MAC. The most common method for building a MAC with a block cipher is the Cipher Block Chaining Message Authentication Code (CBC-MAC). The Algorithm is as followed: [20]

INPUT: data x, secret key k
OUTPUT: m-bit MAC mac
1. Padding and blocking: Pad x if necessary. Divide the padded text into 128-bit blocks denoted $x_1, ...x_t$.
2. CBC processing: enc_k denotes an AES encryption with the key k, compute the block H_t as follows: $H_1 = enc_k(x_1)$;
$H_i = enc_k(H_{i-1} \oplus x_i)$, with $2 \leq i \leq t$.
3. Output generation: Use the function $G : \{0,1\}^{128} \rightarrow \{0.1\}^m$ to generate the MAC $mac = G(H_t)$.
G can be defined as the m most significant bits of H_t.

This algorithm is illustrated in figure 6.6. Note that the keys k are used as AES-keys during the MAC generation while in the generation of the one-way chain, these keys are used as AES-inputs. Hence, it is not possible to gain extra information from the MAC to break the one-way chain nor the other way round as the same keys are not used twice.

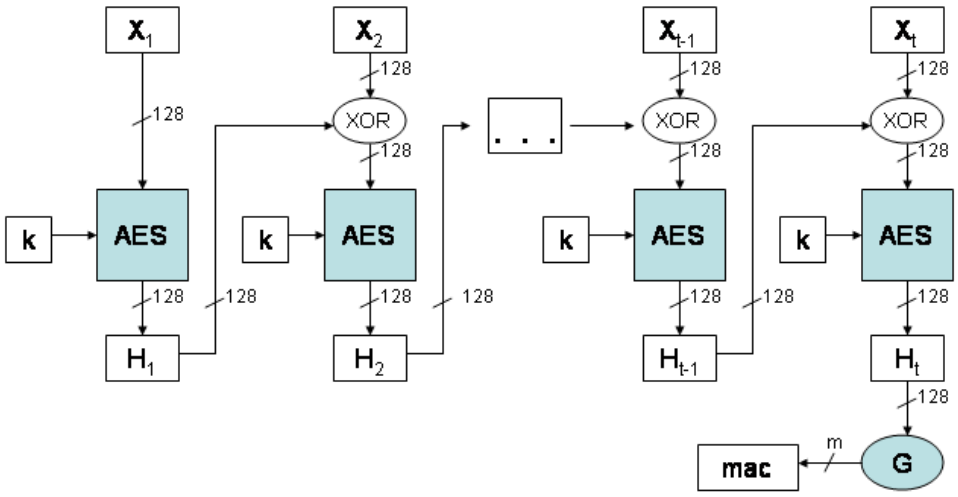


Figure 6.6: The generation of a CBC-MAC.

6.2 The message format and key schedule

This chapter focuses on the tradeoff between authentication time, available bandwidth and security level. In eLoran, one data message consists of a 4 bit header, 41 bit payload and 75 parity bits. The 4 bit header results in 16 different message types. I assume in the following that two of these message types will be reserved for authentication messages. I will refer to them as $type_1$ and $type_2$. With the results of the previous chapters 5 different options are chosen.

Option 1: 80 bit key, no MAC
Only use an 80 bit key to authenticate the time and station, no MAC is used. Only two data messages are needed to send the 80 bit keys. In this case, we will only use $type_1$. The first bit of of the payload of $type_1$ will indicate if it is the first or if it is the second part of the key.
Option 2: 120 bit key, no MAC
Only use a 120 bit key to authenticate the time and station, no MAC is used. Three data messages are needed to send the 120 bit keys. In this case, we will use both, $type_1$ and $type_2$. If the first bit of $type_1$ is 0, the first part of the key has been transmitted, otherwise the second part of the key has been transmitted. In $type_2$ the last part of the key will be transmitted.
Option 3: 80 bit key, 40 bit MAC
Use an 80 bit key to authenticate the time and station and use a 40 bit MAC to authenticate the data messages. Three data messages are needed. $type_1$ will be used to transmit the keys. The first bit will indicate whether it is the first or the second part of the key. $type_2$ will be used to transmit the 40 bit MAC.
Option 4: 120 bit key, 40 bit MAC

Use a 120 bit key to authenticate the time and station and use a 40 bit MAC to authenticate the data. Four data messages are needed. $type_1$ will be used to transmit the first two parts of the key. Again, the first bit will indicate whether it is the first or the second part of the key. If the first bit of $type_2$ is 0, then the third part of the key is transmitted, a 1 indicates the 40 bit MAC.

Option 5: 89 bit key, 32 bit MAC

Use an 89 bit key to authenticate the time and station and use a 32 bit MAC to authenticate the data messages. Three data messages are needed. $type_1$ will be used to transmit the first 80 bits of the key. The first bit will indicate whether it is the first or the second part of the key. $type_2$ will be used to transmit the last 9 bits of the key and the 32 bit MAC.

Which option should be used depends on three factors, how dangerous of counterfeit correction message attacks are, the available authentication bandwidth and the needed security level. The first decision that needs to be made is whether or not a MAC is needed. If some attack scenarios can only be done by forging a data message of the LORAN data channel or if a counterfeit correction message attack is easier than a selective-delay attack, a MAC should be used. Furthermore, if enough bandwidth is available for authentication purposes a MAC should be used as well. But if the authentication bandwidth is very small, one of the two options without a MAC can be chosen. If only the already proposed LORAN messages are used in eLoran, a counterfeit correction message attack will not be easier than a selective-delay or shifting the tracking point attack. Furthermore, every attack that is possible with a counterfeit correction message attack can also be achieved with these two attacks. Therefore, if eLoran is realized as described in [1], it is possible to go without a MAC. Using no MAC will have the advantage that more keys can be sent during the same time. This will reduce the authentication time of the signal a lot.(In case a 80 bit key is used 50% and in case a 120 bit key is used 33%) If the decision is made that no MAC is needed, the choice between Option 1: 80 bit key, no MAC and Option 2, 120 bit key, no MAC depends on whether enough bandwidth is available for a 120 bit key and whether the increased security of a 120 bit key is more important than the decreased authentication time of a 80 bit key.

If the choice is made to use a MAC, e.g. because the LORAN data channel is also used to transmit important warning messages that need to be secured, three different options are available. Option 3: 80 bit key, 40 bit MAC is the option with the highest long-term security. But this option also uses the most bandwidth and should therefore only be used if a lot of bandwidth is available for authentication purpose. The choice between Option 4: 80 bit key, 40 bit MAC and Option 5: 89 bit key, 32 bit MAC depends on the error rate of the data channel. If the data channel is not very stable and data losses occur quite frequently, a 80 bit key should be used. This has the advantage, that the time and source of the signals can be authenticated even if the third message, containing the MAC, is lost. In this way at least the most important information can still be authenticated. If Option 5: 89 bit key, 32 bit MAC is used and the third data message is lost, the attacker will not be able to authenticate any information as the third message contains parts of the key

as well as the MAC. Hence, if the data channel is very stable, this option should be used due to the longer key length, but if the data channel is not very stable, option 4 80 bit key, 40 bit MAC should be used.

Besides the bandwidth and the used key length, the authentication time of the data depends on the used security delay d. The security delay d is defined as the clock offset that will be introduced by the attacker if the attacker uses an already published key to forge a MAC. The bigger the security delay is, the more likely will a receiver detect the attack.
Two different delay scenarios are discussed:
delay option 1: next-key option
The corresponding key for a MAC is the next published key.
delay option 2: key-after-next-key option
The corresponding key for a MAC is the key after the next published key.

The first impression is that the advantage of the key-after-next-key option is the increased security delay. However, the same delay can be achieved with the next-key option by simply reducing the bandwidth used for authentication purpose. This has the advantage that less bandwidth for authentication purpose is used with the same security delay. The advantage of the key-after-next-key option is that during the same amount of time two MACs and key pairs will be transmitted while only one MAC-key pair will be transmitted with the next-key option (if the bandwidth is reduced accordingly). This results in two advantages: 1. In the same amount of time two MACs will be authenticated. If an error occurs during a data transmission, the time until the next MAC will be authenticated is reduced. 2. While the MAC authentication time is the same for both delay options, the authentication time of the current time and station id will be half the time for the key-after-next-key option compared to the next-key option, due to the fact that two, instead of one, one-way chain keys will be transmitted.
The big question is how much security delay is needed. If we increase the delay, the chances that a receiver detects a forgery increases, as it becomes more and more unlikely that the receiver does not detect the clock offset. However, an increased delay also results in an increase of authentication time. We will at first analyze the delay and security level of the next-key option for the case that an 80 bit key is used. In this case, the data which will be signed is sent first, then the MAC mac_i followed by the first half of the key $k_{i,1}$ and finally the second part of the key $k_{i,2}$ is being sent. (With $k_i = k_{i,i}||k_{i,2}$ being the key for mac_i) To be able to authenticate the MAC, the receiver needs the four messages containing mac_i, $k_{i,1}$, $k_{i,2}$ and the data message m. An attacker could try to forge the MAC before receiving the complete key k_i by guessing the remaining bits. If the attacker knows 40 bits of the key, the chance that the attacker guesses the correct key is $1/2^{40}$. The attacker can check his guess by trying to validate his key guess. Testing 2^{40} keys can already be done with special purpose hardware within a couple of seconds. Because of Moore's Law, it will probably be possible in the future to test 2^{40} keys in less than a second. As the Reed Solomon forward error correction is used, the attacker can predict with

a high chance the remaining bits of a message even before the entire message has been transmitted. Hence, the MAC becomes insecure at the moment the first part of the key is being transmitted.
In case a 120 bit key is used, the attacker would need to guess $2^{80}/2$ keys on average after the first 40 bits are revealed. This is computational very difficult. Although in the future it will become computational feasible to test the 2^{80} bits, it will still be extremely difficult to achieve this in real time. To gain an advantage by guessing, the attacker needs to be able to tests these bits before the next part of the key will be revealed, which will be about 30 seconds later. Hence, if a 120 bit key is used, the attacker will be able to forge the MAC only after the start of the transmission of the second part of the key. Figure 6.7 illustrates the security delay and authentication delay for an 80 bit key and a 120 bit key.

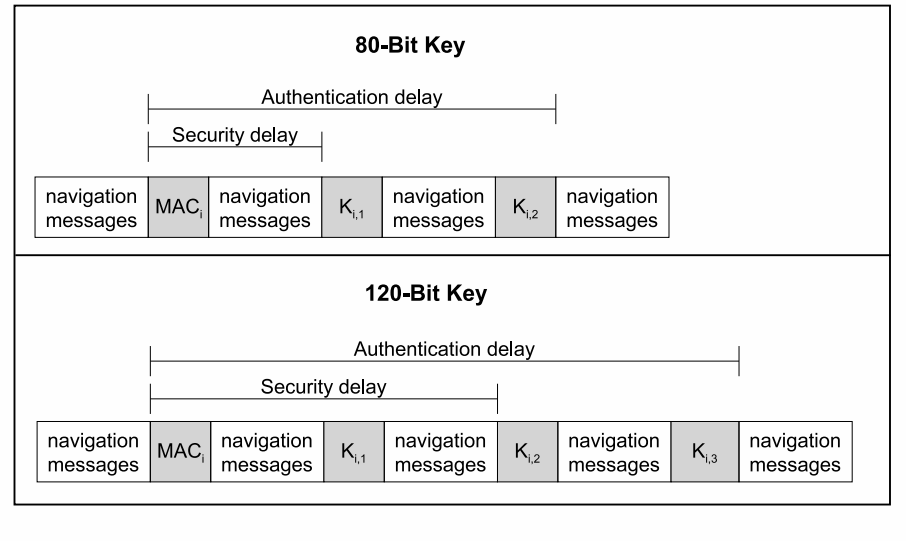


Figure 6.7: The authentication delay and the security delay for an 80 bit and 120 bit key. The corresponding key K_i for MAC_i is $K_i = K_{i,1}||K_{i,2}$ for an 80 bit key and $K_i = K_{i,1}||K_{i,2}||K_{i,3}$ for a 120 bit key.

The transmission time of one eLoran message depends on the GRI of the LORAN chain. The longest GRI has a GRI of 99.9 milliseconds resulting in about one data message every 2.4 seconds. The smallest GRI in the US has a GRI of 59.3 milliseconds resulting in about one data message every 1.4 seconds. The time between two authentication messages depends on how many other eLoran messages are transmitted between these authentication messages. A bandwidth of 10% for authentication messages means that every tenth eLoran message one authentication messages is being send out. As eLoran will need a lot of bandwidth to transmit the correction and time messages, it is reasonable to believe that no more than 7.5 % of bandwidth will be available for authentication messages. It is theoretically also possible to add

a tenth pulse to transmit additional 5 bits. In this case, it might be possible that up to 100% of the bandwidth of this second channel is available for authentication purpose. However, an authentication bandwidth of about 7.5% is more likely.
For a bandwidth of 7.5 % and a GRI of 593 the security-delay is about $1.4s \cdot (1/7.5\%) = 18.6s$ for an 80 bit key and about $1.4s \cdot (1/7.5\%) \cdot 2 = 37.3s$ for a 120 bit key. The authentication delay is $1.4s \cdot (1/7.5\%) \cdot 2 + 1.4s = 38.7s$ for an 80 bit key and $1.4s \cdot (1/7.5\%) \cdot 3 + 1.4s = 41.4s$ for a 120 bit key.
However, for a GRI of 999 and a bandwidth of 7.5%, the security delay is $2.4s \cdot (1/7.5\%) = 32s$ for an 80 bit key and $2.4s \cdot (1/7.5\%)) \cdot 3 = 64s$ for a 120 bit key and the authentication delay is $2.4s \cdot (1/7.5\%) \cdot 2 + 2.4s = 66.4s$ for an 80 bit key and $2.4s \cdot (1/7.5\%) \cdot 3 + 2.4s = 98.4s$.
In practice, not the average or best security delay and authentication time matters, but the worst case scenario. If a security delay of 32 seconds is considered secure it does not matter that this is achieved with a GRI of 9990 if other LORAN chains such as 5930 do not meet this requirement. The entire system would still be considered insecure. The same is true for the authentication time. If an application has an upper bound of the authentication time of 41 seconds, this application would not be feasible if there are chains in which the authentication time is above 41 seconds, regardless whether or not some chains meet the time requirements. The best solution for this problem is to use a different bandwidth according to the GRI. In the following, we assume that for GRIs smaller than 9990 the authentication bandwidth will be reduced, such that these chains have about the same security delay as the 9990 chain. In the following analysis the maximum transmission time of 2.4 seconds for one data message is used. Table 6.4 shows the security delay and authentication delay for the next-key option. Table 6.5 shows the same results for the key-after-next-key option.
The length of the needed security delay depends on the offsets of the used clocks. The offset of a watch crystal is less than 1.7 seconds/day. Hence, a good watch crystal should never have an offset of more than 2 seconds per day over several days. Therefore we can set an upper limit of the offset to two seconds per day over several days. If a bigger offset is reached over several days, this is a very good indicator for an attack as in normal operation such an offset should never be reached. To generate the needed offset in the clock, an attacker has basically three options: In the first option, the attacker delays the LORAN signals to slightly introduce the needed offset. The second option is to physically tamper the clock, for example by changing the operation temperature of the receiver. However, this requires access to the receiver. The third option is to jam the receiver so that the receiver does not receive the correct time for a long time, and wait until the clock offset is big enough due to normal clock drifts.
For an 80 bit key, the attacker needs an clock offset of at least 32 seconds. If we set the upper bound of the clock offset to 2 seconds per day, an attacker needs $32/2 = 16$ days to generate the desired offset by slowly introducing a delay to the LORAN signal. If an offset of 32 seconds is reached in less than 16 days, this is an indication for an attack. For high security applications, oven controlled oscillators with an offset of about 86 microseconds/day or even atomic clocks with offsets of

only 86 nanoseconds per day can be used. To create an offset of 32 seconds to one of these clocks without physical access will be quite impossible. You could easily define that such a clock will only accept on average an offset of 200 microseconds a day, so that 160000 days would be needed to create an offset of 32 seconds without physical access.
If a 120 bit key is used, the security delay is 64 seconds with the result that an offset of 64 seconds is needed. With an upper bound of an offset of 2 seconds a day, an attacker would need about 32 days to create a sufficient offset by delaying or jamming the LORAN signals. However, if physical access if given to create the necessary offset is much easier achieved by tampering the clock, e.g. by changing the temperature. How easy this is depends on the countermeasures against tampering. For high security applications, and scenarios in which the attacker will very likely have physical access, sophisticated anti-tamper methods can be build into the receivers.

In the calculations we assume a bandwidth of 7.5% so that every 32 seconds, or about every 13th eLoran message one authentication message is send. Hence, three authentication message are send within 39 messages. In the computations above we assumed that all authentication messages are evenly spread over the 39 messages, so that the first message is the MAC, message 14 is the first part of the key and message 27 is the second part of the key. If we change this schedule, so that the first message is the MAC, and the 26th message is the first part of the key and the 27th message is the second part of the key, we would have a security delay of 25 messages compared to a security delay of 13 messages. The authentication time is the same, as in both cases the 27th message contains the second part of the key. However, the security delay is increased by $(25 - 13)/13 = 92\%$.
Hence, the security delay does not only depend on the available bandwidth, but also on the used message schedule. The longer the time between the MAC and the first part of the key, the longer the security delay.(In case a 120 bit key is used the time between the MAC and the second part of the key is important) Using this method, the security delay and thereby the security can be increased. However, transmitting more than one authentication message during a short time period might violate other eLoran restrictions, as during this time there might not be enough bandwidth to transmit all necessary time and correction messages.

6.2.1 Using a MAC depending on the need

Besides the two options, to either use a MAC and a key or to use the keys on their own, there is a third possibility, to use a MAC in dependence of the situation. If no MACs are used, a lot of bandwidth can be saved. With this bandwidth other data such as correction messages can be transmitted. It can also be used to transmit the one-way chain keys more frequently. This will reduce the time needed to authenticate the signal significantly. On the other hand, MACs are a powerful protection against attacks that include forging eLoran messages such as counterfeit correction message attacks. If the LORAN data channel is also used to transmit data not directly linked

	80-bit key		120-bit key	
Bandwidth (%)	Auth. delay	security delay	Auth. delay	security delay
100	7.2 s	2.4 s	9.6 s	4.8 s
50	12 s	4.8 s	16.8 s	9.6 s
10	50.4 s	24 s	74.4 s	48 s
7.5	66.4 s	32 s	98.4 s	64 s
5	98.4 s	48 s	146.4 s	96 s
2.5	194.4 s	96 s	290.4 s	192 s
1	482.4 s	240 s	722.4 s	480 s

Table 6.4: Authentication delay and security delay for a MAC if either an 80 bit key or a 120 bit key and delay option 1, the next-key option is used.

	80-bit key		120-bit key	
Bandwidth (%)	Auth. delay	security delay	Auth. delay	security delay
100	14.4 s	9.6 s	19.2 s	14.4 s
50	26.4 s	19.2 s	36 s	28.8 s
10	122.4 s	96 s	170.4 s	144 s
7.5	162.4 s	128 s	226.4 s	192 s
5	242.4 s	192 s	338.4 s	576 s
2.5	482.4 s	384 s	674.4 s	480 s
1	1202.4 s	960 s	1682.4 s	1440 s

Table 6.5: Authentication delay and security delay for a MAC if either an 80 bit key or a 120 bit key and delay option 2, the key-after-next-key option is used.

to eLoran, these data messages might be especially vulnerable. Such messages could include warning message such as hurricane or wildfire warnings. The LORAN data channel might also be used to inform about the status of other navigation systems such as GPS [19]. It might be reasonable to authenticate these messages using a MAC, while it might not be necessary to authenticate normal correction messages. Furthermore, it might also be necessary to authenticate unusual big correction and warning messages as these messages have a big impact on the computed position. In the following, a way to securely switch between using only the one-way chain keys and using a MAC is introduced.

To be able to switch between using and not using a MAC, it must be defined when exactly a MAC will be sent and when not. There are two ways to do that. The first way is to define a schedule at what time a MAC will be sent. For example, every ten minutes a MAC will be sent out, while every minute a one-way chain key will be transmitted. This results in a message authentication time of at least 10 minutes, while the freshness of the signal can be checked every minute. This scenario is very inflexible and it would not be possible to send out MACs quicker for warning messages.
Unfortunately, it is not possible to switch between the use of MACs and the use of no

MACs using only one chain and defining which messages need authentication. For example, it is not enough to define that all correction messages, resulting in offsets greater than 20 meters, need to be authenticated. If the base station sends out a correction resulting in an offset of 50 meters and signs it using a MAC, an attacker could simply forge the messages to corrections of only 2 meters without sending the MAC. As correction messages with offsets of 2 meter would not be signed per definition, a receiver would not expect a MAC and would falsely verify this message as valid. However, this attack can result in an offset of 48 meters or more.
The other option is to use two one-way chains. One chain is used with a MAC and the other chain is used without a MAC. If two one-way chains are used, simply leaving out the MAC is not possible. If the transmitter sends a MAC, it uses a key from the one-way chain assigned to MAC-usage. A receiver does not accept such a key without a MAC. It is not possible for an attacker to send a valid key from the one-way chain assigned to use without a MAC, because such a key for the current timestamp has not been published.
There are two drawbacks in this solution. One drawback are the doubled storage requirements, as two public keys need to be stored for every station. The other drawback is the increased computation time. If the last verified key is very old, a lot of computations need to be done to validate the current key. This normally occurs if either the receiver has been offline for a long time or if the receiver moves to the coverage area of a transmitter where it has not been for a long time. If two one-way chains are used, these computations need to be done twice. Plus, if the MACs are used very seldom, these keys might be especially old. Messages which are signed contain more likely critical information and need to be authenticated very fast. Therefore, MACs should be sent out not only for critical messages, so that the last verified keys of both chains are not too old. For example, it is reasonable to make sure that at least once every hour a MAC is send out, so that both chains are up to date.

7 Further countermeasures

Only signal-synthesis and counterfeit correction message attacks can be prevented using message authentication. Selective-delay and relaying attacks are still possible. Therefore, additional security mechanisms are needed. However, no solution for this problem is known today for terrestrial, low-frequency radio navigation systems such as eLoran. The only known countermeasures against signal-synthesis attacks for global positioning systems are hidden markers or to hide the signal. Both systems use DSSS to hide information in the noise. But it is not possible to use DSSS in low-frequency systems such as eLoran. Furthermore, terrestrial systems have to face the problem that the users and attackers can be as close as several meters from the transmitter station as well as several hundred kilometers away. This makes hiding a signal very difficult. Satellite based systems do not face this problem, as all attackers and users can be expected to be on the earth. Because no methods to prevent signal-synthesis attack for terrestrial, low-frequency systems exist, in the last part of this thesis I try to find possible solutions for this problem.
There are two different approaches that could prevent, or at least hamper, selective-delay attacks. The first approach is to make it impossible to retransmit or delay signals from different stations on their own. Retransmitting all signals with the same delay might still be possible. But if it is not possible to separate each signal, it will not be possible to launch a selective-delay attack. This is the basic idea behind hidden markers (see section 3.4) and pseudo-random signal transmissions. The other approach is to generate different data at different locations. Because the data at the current position and at the wanted spoofing position differs in this case, a selective-delay attack would be impossible. This is the basic idea behind colliding signals.

7.1 Pseudo-random signal transmissions

Each transmitter sends out pseudo-randomly some signals. These signals should be sent at pseudo-random time intervals and with pseudo-random signal strength. The receiver records all of these signals. Some time after the transmission, each transmitter reveals the exact transmission time and signal strength of these signals. This information can be used by the receiver to validate the stored signals. If all signals arrived at the expected time, then the assumed position is valid. The idea behind this countermeasure is that an attacker does not know the source and direction of each signal. But without this knowledge the pseudo-random signals will not match with the navigation signals and the wanted spoofing position, because the attacker does not know the time he needs to delay each signal. For example, if the attacker delays two signals from one station for different amount of times, the receiver will

detect the attack. Therefore, selective-delay attacks would not be possible.
If the signals are sent with the same signal strength, the attacker could use the signal strength of each signal to determine the transmitter station of the signal. Therefore, the used signal strength should vary, so that the attacker can not use this information to determine the source of the transmission.
Besides comparing the signal strength, the attacker could use arrayed antennas to determine the direction of the signals. If the attacker knows the direction of the signal, he also knows the time he needs to delay this signal. Hence, if the attacker is able to determine the direction of the signal, he will be able to perform a selective-delay attack. Because such antennas are easily to get, it is quite easy to attack this system.

But using different signal strengths to transmit the signal can provide additional security. If a weak signal is sent out by a transmitter station, this signal might not be detectable if the distance to the receiver is too big. A signal-propagation model can be used by the receiver to decide which signals should be strong enough at the current position to be detectable. If the distance between the position of the attacker and the wanted spoofing position is big, the attacker might be outside the coverage area of a weak pseudo-random signal, so that the attacker can not receive this signal. However, the wanted spoofing position might be within the coverage area of this signal, so that the receiver expects this signal. As the attacker will not be able to detect and relay this signal, a selective-delay attack becomes impossible. However, the coverage area of a signal depends on various parameters and can not be predicted very precisely. For example, it depends on the surrounding structures and the exact propagation path of the signal. Furthermore, the receiver's equipment and antenna makes a big difference. An attacker with a high-gain antenna is able to recover signals within a much wider area as with the average user equipment. Therefore, this countermeasure is only helpful if the attacker tries to spoof the receiver for huge distances of several dozens of kilometers. The idea of the lost signals is similar to the idea behind colliding signals.
All in all, pseudo-random signal transmissions are only powerful against unsophisticated attackers without directional antennas or if the spoofing position is very far away from the actual position.

7.2 Colliding signals

The basic idea is that the transmitter stations send out data signals that will meet each other (collide) at different locations. If two of these signals arrive at one location at the same time, the two signals will jam each other, so that the data is lost at this location. Several of these signals are sent out with different delays between them to ensure that at each location different data signals are lost. The transmitted data is generated using a pseudo-random number generator with a secret seed value. After all parts of the data stream have been sent out, the seed value is revealed and authenticated. Each receiver can then check whether the received data stream is valid or not, and if no parts of the data stream are missing that should not have been jammed at the current location. An attacker cannot spoof a receiver, due to

the fact that he has lost parts of the data stream that are needed at the wanted spoofing position.

7.2.1 Simulation

Using a continuous data stream:
The station A sends out a pseudo-random data stream S_A. This data is generated using a pseudo-random number generator with a secret seed value. A second station B sends out short jam signals J. While the receiver receives the jam signal J, parts of the data stream are lost. Which part of the data stream is lost depends on the location of the receiver. After the data stream is sent out, the seed value which was used to create the data stream will be published and authenticated. For example, the adjusted TESLA keys can be used as the seed value. In this way, the seed gets published and authenticated without the need of additional bandwidth. The receiver uses the seed to generate the correct data stream and compares it with his received data stream. To verify the position, the receiver checks if only the parts of the data stream are missing that should have been jammed at the current position. At least two different transmitter stations should be used to send out jam signals to get a two dimensional bound of the area in which the same data is lost.
To reduce the area in which the same data is lost, more than one jamming signal data-stream pair should be used. Each jam signal should be shifted in time, so that the area in which the same data gets jammed differs for each data stream, jam signal pair.

Using one station to continuously send out a data stream and one station to send out a jam signal will be very difficult and inefficient in eLoran. This is due to the fact that transmitting data continuously with current eLoran transmitters is very difficult and would require a lot of energy. A more practical approach is to let the stations transmit short pulses of about 50 microseconds containing one or more data bits with delays of 200 microseconds or more between each of these signals. The signal must be designed in a way, so that it is ensured that if two signals meet at the same time, the data will be lost. A Matlab simulation is used to analyze the feasibility of this approach. There are several parameters that have an impact on this system.

`Signal length:`
The signal length depends on the used data modulation technique. On the one hand, it must be ensured that the data can be safely recovered if the signal is not jammed by other stations. On the other hand, it should not be possible to recover the data if signals from two stations arrive at the same time. As LORAN uses a frequency of 100kHz, one pulse is 10 microseconds long. In the following, we assume that five pulses will be used to carry the information, resulting in a signal length of 50 microseconds. Furthermore, we assume that the data is lost if at least 50% of the signals overlap. However, if less than 50% of the signal overlap, the data might still

be lost. The overlapping signal creates interference that might result in a higher error rate. Furthermore, an attacker will very likely use better antennas and signal processing than the normal user, so that the attacker will be able to recover signals that a normal user might loose.
A trade off between false-rejection and false-acception rate is needed to determine how much of the signals need to overlap so that it is ensured that the data is lost, and how much of the signals are allowed to overlap, so that the data can be safely reconstructed. The exact values depend on the used signal structure. The tighter the border between loosing the data and receiving the data can be defined, the more accurate the system becomes. In our simulation we used several different values. In one case we assume that the data is lost if 50% or more of the signal overlap, otherwise the user is assumed to receive the data. This is a very optimisitc assumption. A very pessimistic assumption is that the data is definitly lost if at least 50% of the signal overlap, while the data might be lost if any parts of the signals overlap. This means that the user will not get suspicious if he looses the data packet if for example 10% of the signals overlap. On the other hand, it is assumed that an attacker will be able to recover the signal if 10% of the signals overlap. If 50% of the data overlap, not even an attacker will be able to recover the data.

`Geometry:`
The coverage area for this system is not only limited by the coverage area of the signals. The geometry of the stations has great impact on the coverage area of the system as well, because the angle of arrival of the signals is very important. Assume that station A and station B send out two signals S_A and S_B and that these signals meet at position P. The smaller the angle between the two signals is, the bigger is the area in which the signals overlap. In the extreme case of an angle $\alpha = 0°$, the two signals will propagate in the same direction with the same speed. In this case, the signals will be jammed at all points that lie on this straight line. To make sure that the area in which the same signals get jammed is not too big, the angle between the the two signals should always be $\alpha \geq 90°$. This is always true if the point P lies within the circle defined by S_A and S_B and the diameter $|S_A S_B|$. In the simulation, the geometry of the three LORAN stations Middletown, Fallon and Searchlight from the West Coast Chain are used.

`Length of one transmission period:`
The upper bound of the transmission length is given by the GRI of the LORAN chain. The signals should not interfere with the navigation signals. Therefore, these signals should only be sent out while no navigation signal is being transmitted. The group repetition interval (GRI) defines for every chain how long the master station waits before sending the next LORAN pulse group. After the master station, each slave station will transmit their signals after some individual delay, called emission delay. The time between the transmission of the last slave station and the transmission of the master station can be used to send out the additional signals, without interfering the navigation signals of the LORAN chain. The time between these transmissions depends on the GRI and the number of slave stations. In the

West Coast Chain, the GRI is 99,400 microseconds and the biggest emission delay in this chain is 41,967.30 microseconds for the Searchlight station. It takes about 2,000 microseconds for the Searchlight signal to reach the other stations. To avoid interference with this signal, the first transmission of the security signals should be transmitted at least 2,000 microseconds after the the last navigation signal. For the same reason no signal should be sent out at least 2,000 microseconds before the transmission of the next navigation signal. So there is a time window of about $99,400 - 41,967 - 2,000 - 2,000 = 53,433$ microseconds for sending out additional signals in one GRI in the West Coast chain. However, in this calculation dual rated stations and interference from other chains are not considered. To avoid these interferences, the time window might need to be smaller and during some GRIs it might not be possible to send out these signals at all.
Besides these limitations, there are other aspects that have an impact on the choice of the transmission length. Instead of using one long transmission period, it can be more efficient to use shorter periods and repeat these periods with different offsets.

`Delay:`
Each station will continuously transmit data signals with delays between each signal. A lower bound of the delay between each signal is given by the LORAN transmitter. To continuously transmit data signals without any delay will be quite impossible with current LORAN transmitters as well as very power consuming. A reasonable value for the lower bound is about 200 microseconds. The choice of the delays is very important to ensure that different signals will be jammed at different locations. If two stations use the same delay, then at some places all data signals from these stations will be lost, while at other places non of the signals arrive at the same time and therefore no data is lost at all. The delay also impacts how many signals are sent out and therefore how many signals are lost. A good balance between lost and received signals is needed.
Besides using one fixed delay for each station during one transmission period, two or more different delays can be used for every station. In this simulation, two different delays are used for every station.

`Offset:`
If the same transmission period with the same delay is repeated again, then the same data signals will be lost at the same locations. This creates redundancy, but does not shorten the area in which the same signals are missing. If the transmission period of station A begins with an offset compared to station B, then the area in which one signal is lost will differ from the previous transmission period. Obviously, if two stations have the same offset, their signals will meet at the same places. To create as short areas as possible in which the same data is lost, the difference between the offsets of two stations should be different for each repetition.

The optimal parameters depend on the signal schedule of each chain and the locations of the transmitter stations. They will differ from chain to chain. Finding optimal parameters would go beyond the scope of this paper. To get a feeling for the

approximate performance of this system I will present some results for reasonable parameters. With better choices of the parameters, better results might be possible. Furthermore, the delay and offset parameters also depend on the parameters that define when the data is lost. In the simulation the same offsets and delays are used for the case that the user accepts data losses if more than 50% of the signals overlap, 25% of the signals overlap, or any parts of the signals overlap.

The performance of the system strongly depends on the parameters that define

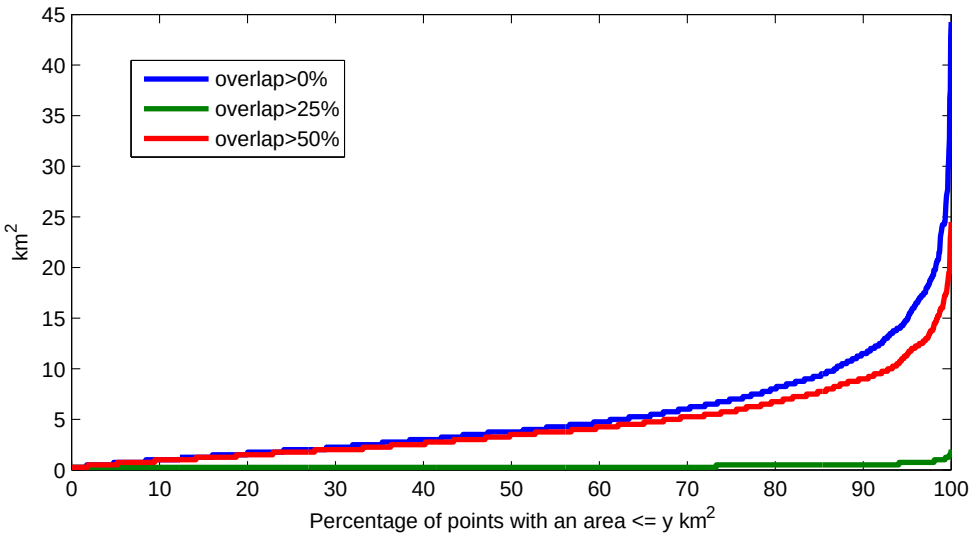


Figure 7.1: The approximate area in which an attacker could use the received data to spoof a receiver. The X-axis represent the percentage of points that have a spoofing area that is equal or smaller than the area shown at the Y-axis. The results are presented for three different cases. The user accepts data losses if more than 50%, 25% or 0% of the signals overlap. For all three cases it is expected that an attacker definetly looses the data if more than 50% of the signals overlap, while the attacker can recover the data if less than 50% of the signals overlab.

how much of the data needs to overlap so that the data is lost. An overview of the performance is given in figure 7.1 and table 7.1. In case the receiver does not accept data losses if less than 50 % of the signals overlap, in 99 % of the testpoints the area in which an attacker would have received all necessary data packets to spoof a receiver is about 1 km^2 or less. On the other hand, if the receivers accept data losses if any part of the signals overlap, this area is about 24.25 km^2 or less. In 95 % of the testpoints the area is about 15 km^2 or less. So attackers could spoof a user within this area. For some applications, the wanted spoofing position will exceed this area so that colliding signals can prevent these spoofing attacks. But for other applications, the wanted spoofing location will be within this area so that a spoofing attack is possible. An area of 1 km^2 on the other hand will significantly increases the attack complexity for most applications, as the wanted spoofing distance probably

Percentage	User accepts data losses if at least x % of the signals overlap.		
	$x \geq 0$ %	$x \geq 25$ %	$x \geq 50$ %
$\leq 99\%$	$\leq 24.25\ km^2$	$\leq 16.25\ km^2$	$\leq 1\ km^2$
$\leq 95\%$	$\leq 15\ km^2$	$\leq 11.5\ km^2$	$\leq 0.75\ km^2$
$\leq 90\%$	$\leq 11.5\ km^2$	$\leq 9\ km^2$	$\leq 0.5\ km^2$
$\leq 75\%$	$\leq 7\ km^2$	$\leq 5.75\ km^2$	$\leq 0.5\ km^2$
$\leq 50\%$	$\leq 3.75\ km^2$	$\leq 3.5\ km^2$	$\leq 0.25\ km^2$
$\leq 25\%$	$\leq 2\ km^2$	$\leq 1.75\ km^2$	$\leq 0.25\ km^2$

Table 7.1: The maximal area in which the same data is lost for an attacker and a user. The user accepts data losses if at least $x\%$ of the signals overlap. In all cases the attacker only looses the data if at least 50% of the signals overlap.

exceeds this area.

7.2.2 Security of colliding signals

The big question is how much the security of the positioning system can be increased with the help of colliding signals. The idea of using colliding signals is to make sure that the receiver receives different data at different locations. In the case of colliding signals, this is achieved by ensuring that at different locations different parts of a data stream are lost. If an attacker tries to spoof the receiver, the attacker needs to know all parts of the data stream that are not supposed to be lost at the wanted spoofing location. As the wanted spoofing position and the attackers position differ, the attacker will have lost parts of the data stream that should not be lost at the wanted spoofing position. Therefore, the attacker will not be able to send all needed parts of the data stream and the receiver will not accept the signals. To launch a successful attack, the attacker has basically two options: The first option is to try to guess the missing parts of the data stream. The second option is to try to gain the missing parts of the data stream using data from more than one place or some kind of advanced signal processing.
The probability that the attacker guesses the correct data stream depends on the number of missing bits the attacker has to guess. The number of bits that will be lost can be increased by repeating the transmission of data streams. The attacker does not have any chance to test his guess and therefore needs to launch the attack without being able to check whether the attack will be successful or not. As the transmission and publication of a data stream takes several seconds, the attacker will only be able to perform a small number of guessing attacks. Therefore, the number of bits the attacker needs to guess does not need to be very big to provide sufficient security. For example, the chance to successful guess only 20 bits is $1/2^{20} = 1/1048576$. On average, an attacker would need to try 1048576 guessing attacks to be successful. This is very unrealistic in most scenarios. If every second one data stream is transmitted, the attacker would need on average about 12 days to successfully guess a single data stream. During all this time, the receiver would

not be able to validate a single signal. This is a very obvious indicator for an attack. If TESLA is used as the seed value for the data stream, a value of one data stream per minute is much more realistic than the one second assumption.
However, in practice a receiver will not always receive all parts of the data stream even if no collision occurred, due to various possible interference. Therefore, the receiver needs to be able to accept a small amount of errors. If the missing or incorrect data occurs always in the same data parts, this can be used as a good indicator for an attack. It is very unlikely that always the same parts of the data stream is lost due to interference. Lets assume that the transmission is stable enough such that not more than 5 bits are normally lost due to interference. The probability to successfully guess 15 out of 20 bits is about 2%. This chance might be high enough for some attackers to launch a succesful attack. So the more bits can be lost due to interference, the more unique[1] bits are needed for every position.
In the simulation about 23 signals are sent out by each transmitter during a single 20 millisecond transmission period. On average 7.6 of these 23 messages are lost. In the simulation ten of these periods are used. Hence, on average 154 signals are transmitted for one unique[2] signal. From these 154 signals on average about 51 signals are lost. Depending on the signal design, one signal can carry more than one bit of data. Figure 7.2 shows the success probability of an attacker for a colliding signals system with 120 unique bits according to the number of bits the attacker is allowed to guess incorrect. How many bits are needed depends on the chance of loosing data bits due to interference and the wanted security level. The smallest amount of unique bits that will still assure a good security level will be about 25 unique bits if 5 bits are allowed to be incorrect. Note that this does not mean that the data loss rate can be as high as 1/5. To generate 25 unique bits much more than 25 bits need to be transmitted. If one signal would only carry one bit of data, in our simulation about 103 received bits are needed for one unique bit in the worst case.
By analyzing the overlapping signals, the attacker might gain a higher chance than 50 % to guess one bit. In this case, the number of unique bits need to be increased to provide equivalent security. Therefore, the same analysis with a propability of 6 % can be seen in figure 7.2 as well.
However, the biggest threat for this security system is not the guessing attack. The biggest threat is an attack in which the attacker collects data from more than one location. If the attacker collects data from two different locations, the attacker might be able to receive the entire data stream, as different data signals are lost at theses locations. Another option the attacker has is to collect the data at the wanted spoofing location. In this case, it is obvious that the attacker will possess all parts of the data stream needed for a spoofing attack. Such an attack is called a relaying attack and there are no countermeasures for passive navigation systems against relaying attacks known today.

[1] A unique bit is defined as a bit that is lost at the current location but that is needed at the wanted spoofing position.

[2] A unique signal is defined as a signal that is lost at the current location but that is needed at the wanted spoofing position.

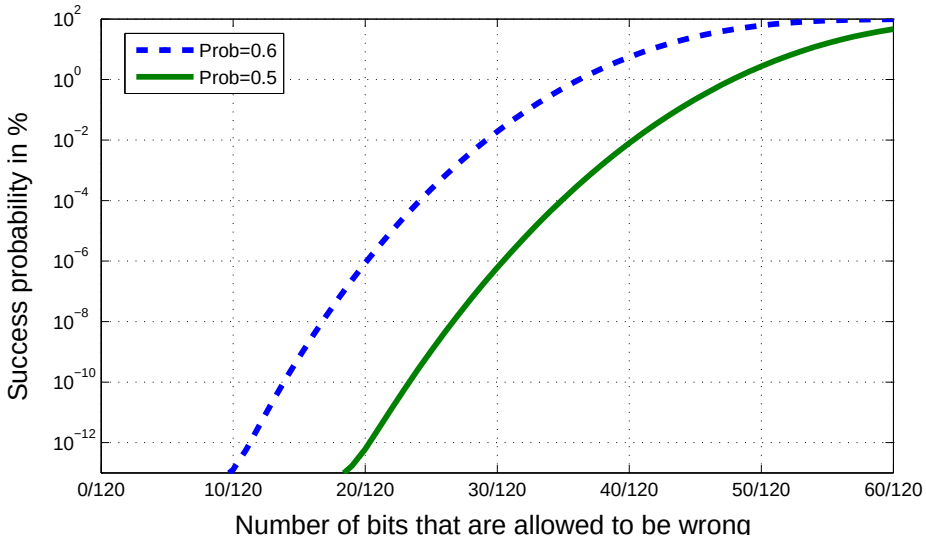


Figure 7.2: The probability that an attacker successfully guesses the needed bits for 120 unique bits. The X-axis defines the number of bits the attacker is allowed to guess incorrect. The red line shows the result in case the attacker has a 50% chance of guessing a bit correct. The dashed blue line shows the results in case the chance to guess one bit is 60% for the attacker.

However, there is one basic drawback with this type of attack. The attacker will need much more infrastructure to be able to collect the data at different locations and to transmit them to the spoofing device. If the attacker collects data at two positions that are too close to each other, then mostly the same data parts are lost at these locations so that not enough of the data stream is revealed to spoof a position that is further away. Therefore, the attacker needs to collect the data at locations that are quite far away from each other (3 km or more) or very close to the wanted spoofing location, making this attack quite complicated.
Besides using more than one receiver to collect data at different locations, the attacker can try to recover the lost data by using high-gain antennas. The signals from each station arrive from different directions. Using high-gain antenna, the attacker might be able to increase the signal gain of one signal and might be able to recover the data. Therefore, separating and recovering each signal might be possible with good equipment.
The chance that an attacker can recover the data depends on the signal strength of the colliding signals and the signal structure. If the signal strength of one signal is much stronger than that of the other signal, this signal can be recovered much easier. Therefore, the security should not rely on loosing both colliding signals. Fortunately, the area in which the same data bits are lost does not become much bigger if it is assumed that only the signal with the lower signal strength will be lost. (In the simulation only one signal is lost if two signals collided, so that the data above already considers this case.) For example, it is possible to define that

only signals that have been sent out from 300 km or further will be lost, as these signals are much weaker than closer signals.
It is also possible, without too much performance losses that only one station transmits data signals. The other stations transmit some kind of jamming signals that are sent with a higher signal power and that are especially designed to jam the data signals. Changing the station, which sends out the data signals, and the ones sending out jam signals, can ensure that the station transmitting the data is not always the closest station. In this way, the chance that the data is lost if the signals collide can be increased significantly.

7.2.3 Feasibility of colliding signals in eLoran

Colliding signals have some nice properties making them look very promising to achieve improved security for terrestrial positioning systems. To find a system in which the receiver expects different data at different locations is very promising, because of the fact that this will prevent selective-delay attacks. Selective-delay attacks are very powerful attacks against eLoran. No technique that can be implemented in eLoran is known today that can defend these attacks. Colliding signals could prevent selective-delay attacks. Unfortunately, there are other attacks that can bypass colliding signals. If an attacker collects data from different locations, he will be able to successfully attack the colliding signals system. However, such an attack will be logistically much more complex so that this type of attack might be impossible for several attack scenarios. Colliding signals might also be attackable using high-gain antennas and other mechanisms that recover the overlapped signals. How hard it is to recover the data depends on several factors, including the signal design, the way the signals overlap, and the used antennas.
Sending out colliding signals requires a lot of additional energy and changes to the transmitter equipment in eLoran. Furthermore, these signals also create interference that might interfere with the navigation signals. Using different frequencies for navigation signals and colliding signals might reduce the level of interference introduced by colliding signals. However, frequency bands are very limited and frequencies close to 100kHz still create some interference at 100kHz. All this makes it very difficult to realize colliding signals in eLoran. Especially the additional interference caused by the colliding signals and the needed changes to the transmitter stations make this system impracticable for eLoran. The security gains that can be achieved by colliding signals do not justify these costs for eLoran. Therefore, colliding signals are unfortunately not the desired tool to increase the security of eLoran against selective-delay attacks.

However, the fact that colliding signals seem to be impracticable for eLoran does not mean that the idea of colliding signals itself is a bad idea. The biggest disadvantage of colliding signals in eLoran comes from the limitations given by eLoran. In other terrestrial positioning systems, a much better performance can be achieved. For

example a higher frequency, longer intervals and more repetitions can significantly increase the performance of the system. Much smaller areas with the same lost data can be achieved in this way. Furthermore, these positioning systems might not suffer the interference problems from these additional signals as LORAN does. Actually, colliding signals could be used on their own as a positioning system, as it is possible to determine the position with the knowledge of the lost signals.
Although colliding signals can be circumvented by attacks in which the attacker can collect data at several locations, it still provides security against attackers that can only collect data at one location. This will be true for many applications. The big advantage of colliding signals is that it is a asymmetric security system and that it is a passive system, as the user does not need to communicate with the transmitter. There are very few alternatives for colliding signals in passive positioning systems which can successfully prevent selective-delay attacks and that do not depend on a shared secret. Hidden markers are one of the alternatives. But they rely on spread spectrum codes and no method of implementing them in terrestrial, high-power, low-frequency systems is known today.

8 Conclusion

In this thesis I discussed the different attacks and countermeasures for positioning systems, focusing on LORAN and eLoran. I showed that attacking eLoran is much more difficult than attacking satellite based positioning systems, due to the high signal strength and the low-frequency. However, LORAN is unprotected to direct-injection attacks, as there are no authentication mechanisms or any other countermeasures implemented in LORAN. For the development of eLoran, I propose the use of a very efficient authentication mechanism to increase the security against signal-synthesis and counterfeit correction message attacks. Because of the very small data rate of only about 17 bits/second, only a very efficient authentication mechanism can be used for eLoran. In this thesis, I developed an authentication mechanism called adjusted TESLA. In adjusted TESLA, the time and station information is linked to the one-way chain generation. This results in two major improvements:
1: The time message works like a counter. I showed that introducing a counter makes attacks on the one-way chain much more complex. An attacker will only be able to compute keys that are valid for a short time.
2: As the time and station information is already embedded in the one-way keys, these one-way keys can be used to authenticate the time and station. This makes it possible to completely leave out the MAC, while still authenticating the most important information.

I showed in section 6.1.1 that by binding the time information into the one-way chain generation, even keysizes of 80-bit will provide reasonable security. Even when brute-force attacks against 80-bit block ciphers become feasible, attacking the one-way chain will still be very difficult and will exceed the capabilities of most attackers. If the attacker can not test 2^{80} keys within a second, the attacker will not be able to attack more than one station at a time. This will limit the attacker in several ways. First of all, the attacker will not be able to launch a selective-delay attack, as at least 4 signals are needed to calculate the exact position. Furthermore, if the attacker only spoofs one signal, this signal will create contradictions if more than 4 signals are available. Hence, the user will be able to detect the attack.
The only advantage an attacker gains by breaking one one-way chain is that the attacker will be able to launch a counterfeit-correction attack. But these attacks are limited, as only errors of +/- 600 meters can be introduced. Furthermore, these correction messages can also be double checked using data from different stations. Hence, an attacker possessing only keys for one chain has only very limited opportunities to use it.
With these results I could show that it is reasonable to use a key size of only 80 bits. For higher security a 120 bit keys can be used. If protection against counterfeit

correction message attacks is needed and enough bandwidth is available for authentication a MAC should be used. My analysis shows that the size of the MAC can be very short to provide sufficient security so that a MAC size of 32 bits provides sufficient security.
To generate the one-way chain efficiently and secure, I propose to use a one-way chain generation based on AES. I also propose to use AES in CBC mode for the MAC generation. The use of AES makes the one-way generation very efficient. With a hardware implementation of the one-way chain generation, it is possible to generate about 2^{24} keys per second or more. In one year about 2^{25} keys are used so that a 30 year old key can be validate in about a minute. Once the receiver has validated a current key, the key validation will only take microseconds, as only about 60 keys need to be authenticated. (If one key pro minute is sent out) This enables the user to validate one-way chain keys even if the public key is very old. In this way, one one-way chain can be used for many years instead of several small one-way chains that need to be authenticated.
In section 6.2 I showed that the keys can be publish directly after the MACs. Because of the very low data rate of the LORAN data channel, the security delay is still big enough in this case. This fact eliminates the disadvantage of TESLA that messages can only be authenticated after a delay. Because the signature size of adjusted TESLA is significantly smaller than that of traditional signature schemes, the authentication delay for adjusted TESLA is much smaller compared to classic signature schemes such as DSA or ECDSA.

Message authentication is only the first step towards a secure positioning system. Message authentication on its own can not be used to defend against selective-delay or relaying attacks. But message authentication is a requirement for many further countermeasures such as hidden markers or colliding signals. In chapter 7, countermeasures against selective-delay attacks for eLoran are examined. Hidden markers, the only asymmetric countermeasure against selective-delay attacks known for GNSS systems, can not be implemented in eLoran, as eLoran uses high-power, low-frequency signals.
In this thesis I proposed a new method to prevent selective-delay attacks, called colliding signals. The idea of colliding signals is to generate different data at different locations, by making sure that different data packets are lost at different locations. The idea is that if two signals arrive at the same time, these signals will jam each other, so that some data is lost. As the lost data differs at each location, this information can be used to detect attacks. In a selective-delay attack the attacker delays each signal for a different amount of time. However, the attacker will have lost data packets different from the ones lost at the wanted spoofing position. Hence, the attacker will not be able to transmit all data packets that the receiver expects to receive.
I used a Matlab simulation to test the performance of the system with realistic parameters for eLoran. The area in which the same signals are lost is quite big (about 3 km^2 - 17 km^2) with the assumed values. Although the values of the simulation were chosen to be realistic for eLoran, implementing this system in eLoran is still

very difficult. Especially the needed change of transmitter equipment and the additional interference caused by the additional signals make the system impracticable for eLoran. The security achievements do not justify these costs.
But although colliding signals are not the desired countermeasure against selective-delay attack for eLoran, they can be very promising for other terrestrial positioning systems. In systems without the limitations given by eLoran, much better performance can be achieved. Furthermore, in other systems, the interference caused by the additional signals might be negligible.

All in all, I showed that the security of positioning systems is very important. Although several ways to achieve better security for positioning systems are known today, there is still a lot of research needed in this area. Several problems remain unsolved, e.g. countermeasures against selective-delay attacks for low-frequency positioning systems such as LORAN or countermeasures against relaying attacks. It is also important to understand the limitations and vulnerability of todays positioning systems. Although more and more applications and infrastructure depends on GPS, civil GPS today is more or less unprotected. This shows the urgent need for better security mechanisms for GNSS systems and the need for an independent and secure backup. eLoran can be a secure and reliable backup for GNSS systems, due to its different errors sources and its robustness against over-the-air attacks. As embedding security mechanisms into existing positioning systems is very difficult and time consuming, the development of eLoran is a great opportunity to embed security mechanisms into the design process of this renewed positioning system.

Bibliography

[1] Enhanced Loran (eLoran) Definition Document. Technical report, International Loran Association, October 2007. Report Version: 1.0.

[2] Draft Specification of the eLoran System. Technical Report RTCM Paper 075-2008-SC127-017, Radio Technical Commission for Maritime Services (RTCM), March 2008. Preliminary, Tentative and Unofficial.

[3] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. NIST SP800-57: Recommendation for Key Management Part 1: General(Revised). Technical report, March 2007.

[4] James Carroll, Karen Van Dyke, John Kraemer, and Charles Rodgers. Vulnerability Assessment of the U.S. Transportation Infrastructure that Relies on GPS. ION National Technical Meeting, Long Beach, CA, January 2001.

[5] J. Clynch, A. Parker, G. Badger, W. Vincent, P. McGill, and R. Adler. Unjamming a Coast Harbor. *GPS World*, January 2003.

[6] Alan Grant, Paul Williams, Nick Ward, and Sally Basker. GPS Jamming and the Impact on Maritime Navigation. In *Journal of Navigation, Volume 62, Issue 02*, pages 173–187, Apr 2009.

[7] T. Güneysu, T. Kasper, M. Novotný, C. Paar, and A. Rupp. Cryptanalysis with COPACOBANA. *IEEE Trans. Computers*, 57(11):1498–1513, 2008.

[8] Gerhard P. Hancke and Markus G. Kuhn. An RFID Distance Bounding Protocol. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 67–73. IEEE Computer Society, 2005.

[9] Yih C. Hu, Markus Jakobsson, and Adrian Perrig. Efficient Constructions for One-way Hash Chains. In *Applied Cryptography and Network Security*, volume Volume 3531, pages 423–441. Springer Berlin / Heidelberg, 2003.

[10] T. Humphreys, B. Ledvina, M. Psiaki, B. O'Hanlon, and P. Kintner. Assessing the Spoofing Threat. *GPS World*, pages 28–38, Jan 2009.

[11] D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.

[12] N. Koblitz. Elliptic curve cryptosystems. In , *Mathematics of Computation, Volume 48*, pages 203–209, 1987.

[13] Markus Kuhn. An Asymmetric Security Mechanism for Navigation Signals. In *In Proceedings of the Information Hiding Workshop*, pages 239–252. Springer, 2004.

[14] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking Ciphers with COPACOBANA A Cost-Optimized Parallel Code Breaker. In *Eighth Intl Workshop Cryptographic Hardware and Embedded Systems (CHES 06)*, pages 101–118. Springer, 2006.

[15] Arjen K. Lenstra. Key lengths. In *Handbook of Information Security, Volume 1: Key Concepts, Infrastructure, Standards and Protocols.* John Wiley & Sons., 2006.

[16] Arjen K. Lenstra and Eric R. Verheul. Selecting Cryptographic Key Sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

[17] Sherman Lo. *Broadcasting GPS Integrity Information Using Loran-C.* PhD thesis, Stanford University, 2002.

[18] Sherman Lo, Benjamin Peterson, and Per Enge. Assessing the Security of a Navigation System: A Case Study using Enhanced Loran. In *Proc. ENC-GNSS*, 2007.

[19] Sherman Lo, Benjamin Peterson, and Per Enge. Using Loran for Broadcast of Integrity Information for Modernized Global Navigation Satellite Systems (GNSS). In *Proc. ENC-GNSS*, 2008.

[20] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 2001.

[21] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in cryptology — CRYPTO '85, volume 218 of Lecture Notes in Computer Science*, page 417. Springer, 1986.

[22] Mitchell J. Narins. Loran's Capability to Mitigate the Impact of a GPS Outage on GPS Position, Navigation, and Time Applications. Prepared for the Federal Aviation Administration, March 2004.

[23] OReilly. Cracking DES. Electronic Frontier Foundation, 1998.

[24] D. Qiu. Security Analysis of Geoencryption: A Case Study Using Loran. In *Proc. ION GPS/GNSS*, pages 1146–1154, 2007.

[25] Di Qui, Sherman Lo, and Per Enge. Security for Insecure Times: Geoenryption with Loran. *GPS World*, November 2007.

[26] Di Qui, Sherman Lo, Benjamin Peterson, and Per Enge. Geoencryption Using Loran. In *Proc. ION-NTM*, 2007.

[27] L. Scott. Anti-Spoofing & Authenticated Signal Architectures for Civil Navigation Systems. In *Proc. ION GPS/GNSS*, pages 1543–1552, 2003.

[28] L. Scott and D.E. Denning. Geo-Encryption: Using GPS to Enhance Data Security. *GPS World*, pages 140–49, April 2003.

[29] L. Scott and D.E. Denning. Location Based Encryption & Its Role In Digital Cinema Distribution. In *Proc. ION GPS/GNSS*, pages 288–297, 2003.

[30] T Stansell. Location Assurance Commentary. *GPS World*, page 19, Jul 2007.

[31] X. Wang, Y. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In *Lecture Notes in Computer Science*, volume 3621, pages 17–36. Springer, November 2005.

List of Figures

List of Tables