AIS Authentication Using Certificateless Cryptography

Axel Rousselot, Nora Cuppens[a] and Samra Bouakkaz
Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Montreal, Canada

Keywords: Authentication Protocol, Automatic Identification System (AIS), Certificateless Cryptography, Cybersecurity, Digital Signature, Maritime System.

Abstract: The Automatic Identification System (AIS) is a maritime situational awareness system, designed as a collision avoidance tool to increase security at sea. Widely accepted, its data is now used for various applications, from maritime traffic predictions to the environmental effects of noise pollution. However, the AIS has been designed without security in mind and does not embed any authentication strategy. Research has shown how this lack of authentication could lead to disastrous consequences. Authentication AIS is thus an active research field, but the constraints imposed by the AIS network require subtle protocol design and careful use of new cryptographic technologies. This work proposes an authentication scheme for the AIS using the advantages of certificateless cryptography. The scheme is backward-compatible with standard AIS versions. We evaluate the performance and security of our proposed scheme through simulations and theoretical analysis. Our results show that our scheme provides strong security guarantees and efficient performance, making it a promising candidate for authenticating AIS signals in practice.

1 INTRODUCTION

Developed during the nineties to improve situational awareness at sea, the AIS was so successful that in 2002, it was made mandatory by the International Maritime Organisation (IMO) for ships of 300 or more gross tons (GT), vessels carrying 150 people or more, and commercial vessels of 65 feet and more.

AIS transmissions allow devices to transmit static and dynamic data, including their Maritime Mobile Service Identity (MMSI), location, or speed. Upon reception, the data is overlaid on the Electronic Chart Display Information System (ECDIS), giving the navigation crew a good overview of the surroundings. Over the past decade, AIS data has started to gain a lot of interest in the maritime and scientific communities. The AIS data is aggregated and published in real-time on public websites, and it is collected into massive databases of global maritime traffic. This enabled scientists to forecast maritime traffic and vessel trajectories (Last et al., 2014a), assess the sound exposure of coastal species (Merchant et al., 2012), or track oil spills (Schwehr and McGillivary, 2007).

Without considering security, the AIS has been designed, leaving the AIS messages unauthenticated. It was not an issue when spoofing radio signals was very costly, but the rise of relatively cheap Software-Defined Radio (Balduzzi et al., 2014). As demonstrated by Balduzzi et al. (Balduzzi et al., 2014), AIS is vulnerable to numerous spoofing attacks that could have disastrous consequences. In this paper, we provide an authentication extension to the AIS to mitigate spoofing attacks. The remainder of this paper is organized as follows: Section 2 introduces the technical background necessary to understand this work. Section 3 then examines the related work as well as the AIS's existing authentication propositions. Section 4 exposes the architecture and the different phases of our authentication proposition, and Section 5 shows the proof of concept we implemented to conduct various experiments and validate our proposition. Finally, Section 6 presents our results, analyses our work's limitations, and explores future work possibilities.

2 TECHNICAL BACKGROUND

This section aims at providing sufficient technical background about the AIS and CLC (Certificateless Cryptography) to understand the different sections.

[a] https://orcid.org/0000-0001-8792-0413

Rousselot, A., Cuppens, N. and Bouakkaz, S.
AIS Authentication Using Certificateless Cryptography.
DOI: 10.5220/0012054600003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 49-61
ISBN: 978-989-758-666-8; ISSN: 2184-7711

2.1 Automatic Identification System

The AIS uses the two maritime reserved Very High Frequency (VHF) channels 87B (161.975 MHz) and 88B (162.025 MHz). As described in International Telecommunication Union Recommendation M.1371-5 (Series, 2014), each channel is divided into 1-minute long frames, and each frame is divided into 2250 26.67 ms long slots, allowing for the transmission of 256 bits at 9600 bauds. There are 27 different types of AIS messages, each providing different pieces of information. These types, along with their purposes, are presented in Table 1. To be transmitted,

Table 1: All 27 types of AIS messages.

#	Message name
1	Position report (scheduled)
2	Position report (assigned scheduled)
3	Position report (response to interrogation)
4	Base station report
5	Static and voyage-related data
6	Binary addressed message
7	Binary acknowledgment
8	Binary broadcast message
9	Standard SAR aircraft position report
10	UTC/date inquiry
11	UTC/date response
12	Addressed safety-related message
13	Safety-related acknowledgment
14	Safety-related broadcast message
15	Interrogation
16	Assignment mode command
17	DGNSS broadcast binary message
18	Standard Class B equipment position report
19	Extended Class B equipment position report
20	Data link management message
21	Aids-to-navigation report
22	Channel management
23	Group assignment command
24	Static data report
25	Single slot binary message
26	Multiple slot binary message
27	Position report for long-range applications

an AIS message can take up to five transmission slots when sent by a class A sender, and up to three when sent by a class B device. The figure shows the formatting of a type 1 AIS message. This message takes only 168 bits, the maximum that can be transmitted in a single emission slot. If the message is longer, such as message type 8 in some cases, the formatting is changed as shown in Figure 1. The AIS footer and header are only sent once, allowing more data to be transmitted than with two separated single-slot emissions. The table shows how many bytes Message 8 embeds in its payload field across multiple slots (Series, 2014).

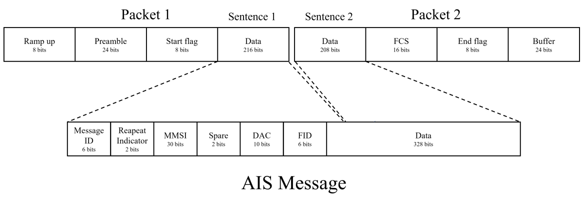


Figure 1: Multislot message AIS formatting example.

3 RELATED WORK

The security of AIS has been widely analyzed in multiple pieces of research. These studies identify the lack of authentication as being the most critical AIS vulnerability, allowing an attacker to easily perform spoofing attacks, denial of service attacks, and hijacking attacks. Over the past decade, various security improvements have been proposed for the AIS. The authors in (Hall et al., 2015) proposed a signature and encryption scheme based on pseudonyms to provide AIS data confidentiality, anonymity, and authenticity. However, the authors did not provide sufficient details about the cryptographic solution they used, how they intend to solve key distribution and management, how much overhead the introduction of their signatures produces, or how they intend to make their solution retro-compatible. The authors in (Su et al., 2017) suggested an authentication scheme based on Rivest–Shamir–Adleman (RSA) signatures along with Certificate Authorities (CA) to provide authenticity to AIS messages. However, as pointed out by the authors of (Goudossis and Katsikas, 2019), RSA produces much larger signatures that can be transmitted with AIS. More recently, the authors in (Wimpenny et al., 2022) suggest using a CA-based scheme and transmitting Elliptic Curve Digital Signature Algorithm (ECDSA) signatures through the VHF Data Exchange System (VDES), the AIS successor. However, the VDES is still under heavy development and is not expected to take over AIS soon. The authors in (Goudosis and Katsikas, 2020) used Identity Based Cryptography (IBC) to authenticate AIS messages. However, IBC is considered insecure because it relies on the use of a trusted third party, known as the private key generator (PKG), to generate private keys for users based on their identities. This means that the PKG has the ability to generate private keys for any user, including themselves, and can therefore impersonate any user (Xiong et al., 2016). Finally, the author in (Nguyen, 2020) suggests using certificateless cryptography to provide AIS message authentication. The author highlights the benefits of using certificateless cryptography over other forms of cryptography but does not provide any implementation of such a scheme.

Several solutions have already been proposed to overcome the lack of authentication in AIS, but no standard has been established. This may be because these solutions are not fully implemented and do not fulfill all requirements. To determine the appropriate type of cryptography, the system's constraints must first be considered, especially the number of transmission slots per minute. The size of the signature is crucial for maintaining the network's integrity, also the infrastructure's complexity and cost must be taken into account. We decided not to use a PKI due to the challenges involved in establishing the infrastructure and transmitting costly certificate chains. Although IBS has some advantages, the complex issue of public key revocation and key escrow is too much to handle. Instead, we chose to adopt CLS to sign AIS messages since it generates smaller signatures, requires less resources, and has lower computational costs (Du et al., 2020).

4 THE AUTHENTICATION PROTOCOL

This section describes our authentication proposition for the AIS.

4.1 Security Issues

Attacks on AIS can affect six significant information characteristics: confidentiality, integrity, availability, possession, authenticity, and utility. Confidentiality involves keeping information secure; integrity involves maintaining the accuracy of information; and availability refers to the accessibility of information. Possession refers to the loss of access to information by authorized users; authenticity involves verifying the identity of the sender; and utility refers to the usefulness of information to the user. After reviewing the existing proposals, we concluded that a solution must meet the following criteria:

- Backward compatibility: The introduction of a solution must not change even partially the behavior of former users or the current standard.
- Standard compliance: Solutions must meet current standards such as NMEA-0183 and ITU-R M. 1371-5 to avoid data transfer size restrictions.
- AIS bandwidth savings: Solutions must minimize transmission of authentication information.
- Internet independence: The solution may still involve ad-hoc internet access for devices that need to check signatures since internet access is easily available at the port.

The confidentiality of AIS data would make it too risky for hacking and interfere with collision avoidance, so we have not included it in these constraints.

4.2 Design Decisions

This section describes the design decisions developed to create a protocol that meets the previously identified design constraints.

4.2.1 Signature Transmission

among the 27 types of AIS messages, three seem suited for signature transmission. The type 6 message, called a Binary Addressed Message, allows an arbitrary binary payload to be sent to a target identified by its MMSI. Its payload has a maximum size of 920 bits using 5 transmission slots, and 496 using 3 slots. Similar to type 6, the type 8 message (Binary Broadcast Message) allows a binary payload of up to 952 bits to be broadcast to all devices in range. Finally, message type 26 (Multiple Slot Binary Message) can embed up to 1004 bits, but it is rarely used in practice, and its use for very specific purposes may cause compatibility issues. Thus, we decided to use the type 8 AIS message to transmit our signatures.

4.2.2 Partial Signing

The number of emission slots necessary to transmit a payload given its size using message 8 is given in Table 2. We excluded the DAC and FID fields of the message, as their usage is reserved for transmitting the message and thus cannot be used for transmitting a signature. To date, no CLC algorithm is able to generate signatures smaller than 41 bytes, so signatures will take at least three emission slots.

Table 2: Given the size of the binary payload, the number of slots used by message 8 to transmit it (excluding the DAC and FID fields).

Number of slots	Maximum binary data bytes
1	10
2	38
3	66
4	94
5	191

If we try to sign every message, the AIS traffic will be approximately multiplied by four (given that most messages are single slots), and this would cause the AIS network to collapse pretty fast. Multiple solutions have emerged from research to tackle this issue. The authors of (Struck and Stoppe, 2021) proposed to sign messages in batches. While solving the issue, this solution may introduce potential reliability problems. Indeed, it only takes one wrongly received bit for the entire message batch to fail verification, and AIS does not implement an error correction algorithm to prevent that from happening. Another solution is to sign only a fraction of AIS messages. To that purpose, we divided AIS messages into two categories. Non-critical AIS messages are position reports and static voyage data sent periodically. Those messages should only be signed once for each S_e message sent by each AIS emitter, where S_e is a security parameter that must be determined. The other messages are considered critical, and should always be signed. We identified messages of types 6 through 14, 16, 21, 22, 23, 25, 26, and 27 as critical. The other ones are considered non-critical. The author of (Last et al., 2014b) determined that non-critical messages represent about 97.35% of all AIS messages. An analysis of an AIS sample available on the *aishub* platform confirmed this number. If we assume that most messages only use one emission slot (which is usually the case for non-critical messages), then the overload factor α is given by the equation 1, where p is the proportion of non-critical messages and S_e is the security parameter.

$$\alpha = 4 - 3p(1 - \frac{1}{S_e}) \tag{1}$$

This means that if the AIS network load is L_i before authentication, the network load after signing is $L_f = \alpha L_i$, where the load ratio between used and unused emission slots. Table 3 shows the final load L_f given S_e and L_i for $p = 0.9735$.

Table 3: Final AIS load given initial load and S_e, for $p = 0.9735$. Red cells are above 100%.

$L_i(\%)$	S_e					
	50	30	20	15	10	5
	α					
	1,14	1,18	1,23	1,27	1,37	1,66
	$L_f(\%)$					
1	1,14	1,18	1,23	1,27	1,37	1,66
12	13,65	14,12	14,71	15,29	16,46	19,96
23	26,17	27,07	28,19	29,31	31,55	38,26
34	38,69	40,01	41,67	43,32	46,63	56,56
45	51,21	52,96	55,15	57,34	61,72	74,86
56	63,72	65,90	68,63	71,36	76,81	93,16
67	76,24	78,85	82,11	85,37	91,89	111,46
78	88,76	91,79	95,59	99,39	106,98	129,76
89	101,27	104,74	109,07	113,40	122,07	148,06
100	113,79	117,69	122,55	127,42	137,16	166,36

This table shows that a value of $S_e = 10$ should be suited for most cases, even though a very crowded area should probably use a higher value to guarantee AIS network integrity.

4.2.3 Signature and Message Linking

Since signatures are sent separately from their messages, we need to find a way to link a signature to its message. We decided to compute a message identifier for each message using the first four bytes of its SHA-1 hash. This 4 bytes identifier is sent with each signature. Thus, when a user receives a message, it calculates its identifier and saves it in a buffer. Upon receiving the corresponding signature, the user can retrieve the message and verify its authenticity.

4.2.4 Replay Attack Protection

To prevent replay attacks on our authentication scheme, we include a 32-bit UNIX timestamp t in the computation of a message's signature σ, as shown in Equation 2.

$$\sigma = sign(SHA1(m|t)) \tag{2}$$

This timestamp will be sent along with the signature and the message identifier to allow receivers to verify that the signature has not been generated more than a security threshold t_h before its reception.

4.2.5 Public Key Repository

A downside of CLC compared to IBC is that public keys cannot be directly derived from a user's identity and thus have to be sent along with a signature. We already assessed that the AIS network was just flexible enough to allow the sending of a few signatures, but could not stand the systematic sending of public keys. Therefore, we decided to use a public key repository, maintained by the KGC. This public key repository contains all the public keys of all registered users and should be downloaded over the internet while ships are in port. Thanks to the cryptographic properties of CLC, this directory does not create a security hole. Even if an attacker were to compromise and modify it, he would not be able to impersonate a user by replacing a user's key with his own or pretending to be a legitimate user by inserting his public key.

4.3 Solution Architecture

Several actors are involved in our solution. They are differentiated into two categories: the administrators, who do not use the protocol directly but participate in its operation, and the users of the protocol. The administrators are divided as follows:

- The lead organization (LO) This entity acts as the KGC's administrative center. It verifies that requests are correct and that an already-registered MMSI is not the subject of new requests. As with KGC, the LO creates partial keys for legitimate users. It is also in charge of maintaining the public key repository and the revocation list.
- Intermediate Organizations (IOs). These entities, located in the various participating countries, are intended to verify the legitimacy of registration requests. Their role is to reduce the administrative load of the LO. They must, under local legislation,

verify the identity of the applicant. This role can be played, for example, by Transport Canada in Canada.

The users of the solution can be divided into three categories:

- Transponders, which can transmit and receive AIS signals, are the most common devices found on ships.
- Transmitters are those devices that can only transmit AIS signals. This category includes, for example, buoys or MOBs. The signature check does not affect these devices because they do not receive AIS signals.
- Receivers are devices that can only receive AIS signals. This is the case, for example, with devices installed on small boats because they are less expensive than transponders. These devices are not concerned with the signature part of the protocol because they cannot transmit. So they don't need to register with the administration to get a key pair, but only to download the public directories to check the signatures.

The overall architecture of the solution is described in Figure 2.

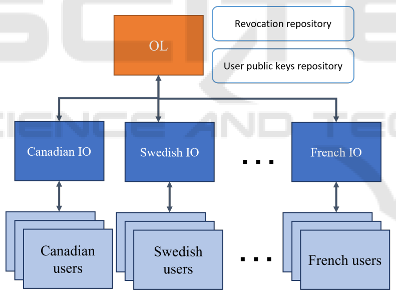


Figure 2: Overall architecture of the solution.

4.4 Operation Phases

This section details how the actors interact during the different phases of the protocol. There are three main phases. The initialization phase, the registration phase, during which a user obtains his keys, and the operational phase, during which a user interacts with other users.

4.4.1 Requirements

For the proper functioning of the solution, we assume the existence of the actors described in the previous section. We also assume the existence of the following means of communication between the actors:

- To enable administrative actions, users and the LO in each country can communicate via email or a governmental web platform.
- Users and the LO can communicate via a web platform provided by the LO. This platform must use the HTTPS protocol so that communications are encrypted.
- Users can communicate with each other by radio, following the AIS specifications.

4.4.2 Phase 0: Initialisation

During this phase, the LO generates the cryptographic parameters of the system as described. First, q is chosen to be an l-bit prime number, where l is the system's security parameter. G is an additive elliptic curve group of order q, and P is a generator of G. The LO chooses $x \in Z_q^*$ randomly and computes $P_{pub} = xP \in G$. x is kept secret as the private master key.n The LO chooses three distinct hash functions: $H_1, H_2, H_3 : \{0,1\}^* \rightarrow Z_q^*$. The public parameters $params = \{q, G, P, P_{pub}, H_1, H_2, H_3\}$ are published.

4.4.3 Phase 1: Registration

This phase is only to be performed by users who can transmit (transmitters and transponders). For a user A, the steps are as follows:

- **Step 1.A.** The user A completes the administrative steps required for registration with their intermediary organization. These steps must verify the identity of the user, as well as the legitimacy of the request. They are specific to each country because they depend on local legislation.
- **Step 1.B.** The intermediary organization checks the request, and if it is compliant, it allows A to set up two-factor authentication on the LO's web platform. The request is forwarded to the LO.
- **Step 1.C.** The LO checks that no other request with the MMSI has been made before , and then it generates a partial key for A based on its identity ID. To do this, the LO randomly chooses $\beta \in Z_q^*$, and computes the same remark above about using the same name β for several variables
  - $y_{ID} = \beta P \in G$
  - $h_1 = H_1(ID, y_{ID}, P, P_{pub}) \in Z_q^*$,
  - $d_{ID} = \beta + h_1 x \in Z_q^*$.

  The partial key is then the couple $D_{ID} = (d_{ID}, y_{ID})$. The partial key is made available to A on the web platform, which he retrieves using the two-factor authentication initialized in step 1.B. All communications are encrypted using the HTTPS standard.

- **Step 1.D.** When user A obtains the partial key, he configures his AIS transmitter with it. The device calculates a public and a private key and presents the public key to A. To compute its keys, the device randomly chooses $v_{ID} \in Z_q^*$ as its private key and computes $g_{ID} = v_{ID}P$. Its public key is then $PK_{ID} = (g_{ID}, y_{ID})$.
- **Step 1.E.** User A, again applying two-factor authentication, sends his public key to the LO. The LO finally updates the user key directory with the public key.

4.4.4 Phase 2: Operational Phase

This phase is divided into three parts. The initialization part requires an internet connection and is only necessary for devices that can transmit. The sending part describes how all transmitting devices should sign messages. The receiving part describes how a device receiving AIS messages should verify the signatures.

Step 2.1: Initialization. During this step, devices must have an internet connection. Two directories are downloaded from a web interface provided by the AIS. A public key directory is a dictionary that contains $MMSI \rightarrow public\_key$ pairs. The revocation directory is a list of revoked public keys. This phase is only necessary for devices that can receive AIS messages because the directories are only useful for message verification.

Step 2.2: Emission. Only a fraction of the AIS messages are signed to avoid overloading the AIS network. The messages considered critical, which are always signed, are those of types 6 to 14, 16, 21, 22, 23, 25, 26, and 27. The other messages are classified as non-critical and are signed only once for each S_e message, where S_e is a system security parameter related to the proportion of non-critically signed messages in $alpha$ by Equation (3).

$$S_e = \frac{1}{\alpha} - 1 \tag{3}$$

A message m must be signed if one of the three following conditions is satisfied:

- The message is a critical one.
- m is the device's first non-critical message since it was turned on.
- At least S_e non-critical messages were sent without a signature.

When a message m needs to be signed, a digest is first computed by applying the SHA-1 hash function to the message m and the current 32-bit UNIX timestamp t, as described in Equation (4).

$$h = SHA1(m|t) \tag{4}$$

Then, the device signs this digest thanks to the *sign* algorithm described by (Du et al., 2020) in the following way: Given the digest to sign h, the user keys D_{ID}, v_{ID} and PK_{ID}, the device computes the following:

- $k \in Z_q^*$ randomly and $\delta = kP \in G$,
- $h_2 = H_2(h, \delta, ID, PK_{ID}, P_{pub}) \in Z_q^*$,
- $h_3 = H_3(h, \delta, ID, PK_{ID}, h_2) \in Z_q^*$,
- $z = k^{-1}(h_2 v_{ID} + h_3 d_{ID}) mod q \in Z_q^*$.

The signature of h is then $\sigma = (\delta, z)$. The message identifier ID_m is calculated from the first four bytes of the message digest using the equation (5). It will allow users who receive the signature to link it to the associated message since the message signature is sent in a separate AIS message.

$$ID_m = SHA1(m)[0:4] \tag{5}$$

Finally, using a type 8 message, the user sends the payload $P = \sigma|ID_m|t$, with a size of $41 + 8 + 4 = 53$ bytes, and thus using only three transmission slots. The DAC used is 100, and the FID is 0.

Step 2.3: Reception. Received messages are divided into four non-distinct categories (a message can belong to none, one, or more categories):

- Critical messages. A critical message belongs to the types mentioned in step 2.2.
- Signatures. A signature is a message of type 8, having a DAC of 100 and an FID of 0.
- Public-key request. A public key request is a message of type 6, having a DAC of 100 and an FID of 1.
- Public key. A public key is a message of type 8, having DAC 100 and FID 2.

The values of the DAC and FID fields adopted for the protocol are arbitrary and will have to be standardized when such a protocol is adopted.

The behavior of the receiving device varies according to the categories to which the received message belongs:

- Case 1: The message was automatically accepted. A message is automatically accepted as a legitimate one if it is non-critical, and the sender is authenticated. A sender user is authenticated when a valid signature for a non-critical message has been received from that sender less than S_e non-critical messages ago. The count of the different types of messages received from different users is kept by the receiving device.

- Case 2: A Message requiring a signature. If the message is critical or the sender is not authenticated, it must be saved until a valid signature is obtained, with the 32-bit UNIX timestamp t_r corresponding to the time of its reception.
- Case 3: Signature. On receiving a signature $s = \sigma|ID_m|t$, from an MMSI ID, signing a message m, the device must first find the message m such that $ID_m = SHA1(m)[0:4]$ and the MMSI of m is ID, the MMSI that sent the signature s. If no message exists, the signature is ignored. If several messages exist, the verification procedure is applied to all of these messages. Then, the time between the transmission and the reception of m is checked. If the message m was received at t_r, we must have $t \leq t_r$ and $t_r - t \leq t_h$, where t_h is a security parameter. If one of these conditions is not verified, the message is rejected because the time between the signature of the message and its reception is too long. Finally, the signature is verified thanks to the $vrfy$ algorithm described by (Du et al., 2020) as follows. Given an AIS message m, a signature $\sigma = (\delta, z)$, the time stamp of the message t, the identity of the sender ID, and the sender's public key PK_{ID}, the device computes
  - $h = SHA1(m|t)$
  - $h_1 = H_1(ID, y_{ID}, P, P_{Pub}) \in Z_q^*$,
  - $h_2 = H_2(h, \delta, ID, PK_{ID}, P_{pub}) \in Z_q^*$,
  - $h_3 = H_3(h, \delta, ID, PK_{ID}, h_2) \in Z_q^*$.

  The message is accepted if $z\delta = h_2 g_{ID} + h_3(y_{ID} + h_1 P_{pub})$.

  Proof. **Correctness: Verify Signature**
  $z\delta = k^{-1}(h_2 v_{ID} + h_3 d_{ID})kP$
  $z\delta = h_2 v_{ID} P + h_3 d_{ID} P$
  $z\delta = h_2 g_{ID} + h_3(\beta P + h_1 x P)$
  $z\delta = h_2 g_{ID} + h_3(y_{ID} + h_1 P_{pub})$ $\square$

  If the key PK_{ID} is not present in the directory, and the device can send AIS messages, a public key request is sent to ID using a type 6 message with a value of 100, 1, and a zero payload. m' and m are set aside while waiting for ID's response. If the device cannot transmit, the signature and message are ignored.
- Case 4: Public key request. If the device cannot transmit, the message is simply ignored. Otherwise, the device must verify that it is the object of the request by comparing its MMSI with the value of the "recipient MMSI" field in the message. If the MMSI matches, the public key is sent thanks to a type 8 message with a DAC of 100 and an FID of 2.

A sequence diagram of the sending of a message and its signature by a user A is presented in Figure 3. In this diagram, users A and B have keys (sk_a, pk_a) and (sk_b, pk_b), identities ID_a and ID_b, and are both transponders. At $t = t_1$, A sends the message m_1. The sending of A's identity is not shown on the diagram because its MMSI is sent in all AIS messages. On receipt of m_1 at $t = t_r$, B internally calculates the message identifier $ID_1 = SHA1(m1)[0:4]$ and stores the message, its reception timestamp t_r and its identifier in a buffer. When B receives the message s, which contains t_1, σ the signature of $SHA1(t_1|m_1)$ by A and the identifier ID_1, B finds the message m_1 thanks to its identifier. B requests A's public key after the temporal checks because it does not have it in its directory. Finally, A answers the request with its key pk_a, and B can perform the signature verification of m_1.

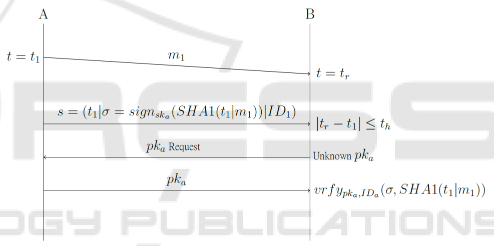

A
B
$t = t_1$
m_1
$t = t_r$
$s = (t_1|\sigma = sign_{sk_a}(SHA1(t_1|m_1))|ID_1)$
$|t_r - t_1| \leq t_h$
pk_a Request
Unknown pk_a
pk_a
$vrfy_{pk_a, ID_a}(\sigma, SHA1(t_1|m_1))$

Figure 3: Authentication sequence diagram of message m_1 sent by user A.

5 SECURITY ANALYSIS

This section demonstrates the proposed protocol's security. Each phase of the protocol's security is analyzed by considering an adversary who can intercept and manipulate radio traffic.

Public Directory Security: Consider the case where an attacker E, possessing an ID_E identity and valid keys (p_E, s_E) generated by OL, succeeds in replacing the public key p_A of a user A in the directory with his own. The attacker's goal is to usurp the identity of A. To do this, E creates a message m containing the MMSI ID_E signs it, using his private key s_E : $\sigma = sign(m, s_E)$, and sends it to user B. B receives m and σ, and retrieves from the directory what it thinks is A's public key to verify the message. It retrieves the p_E key. The verification is carried out as follows: verify (m, σ, p_E, ID_A). This verification fails because the identity used to generate the p_E key is not the same as the identity used in the verification. Even if an attacker convinces other users to use his public key for verification, he cannot sign a message on behalf of someone. This feature is only found in signatures without certificates. During the initialization phase, the OL generates the public parameters without any communication from the actors. The cryptographic protocol proposed by (Du et al., 2020) ensures the security of this phase.

Phase 1: Although the administrative steps in Phases $1.A$ and $1.B$ could be vulnerable to a social engineering attack where an attacker requests on behalf of another legitimate user, this risk is generated by administrative steps that are not part of the protocol. in this analysis, administrative steps are therefore assumed to be secure. The cryptographic security of step $1.C$ is equivalent to the security of the HTTPS protocol. Indeed, if an attacker could decrypt the user's partial key, he could use this partial key to generate a valid pair of keys. Two-factor authentication helps protect this phase of the protocol against a phishing attack where an attacker would send a fake partial key download link to retrieve a user's login password. During step $1.D$, the user generates his private key and public key from his partial key. The cryptographic protocol proposed by (Du et al., 2020) ensures its security. Finally, the security of step $1.E$ is ensured by the HTTPS protocol and two-factor authentication. The public key will be freely available to all users, so an attacker who can eavesdrop on communications has no advantage. Even if the attacker can modify the traffic, he could at most do a temporary denial of service against a single user, but this attack would be very quickly noticed and easily remedied.

Phase 2: This paragraph studies the security of the operational phase of the protocol. Two attacks are studied, the replay attack and the key hijacking attack. The replay attack scenario is as follows. An attacker equipped with an *SDR* records a message m sent at time t by a user A and his signature $s = \sigma \mid t \mid ID_m$, with $\sigma = sign(SHA1(m \mid t), s_A)$. Later, at a time $t\prime > t_h$, the attacker transmits thanks to his $SDR m\prime = m$ and $s\prime = \sigma \mid t\prime \mid ID_h$. Upon receipt of $m\prime$ and $s\prime$ at time t_r, user B first verifies that $t_r - t\prime \leq t_h$. Then it computes verify $(\sigma, SHA1(m \mid t\prime), p_A, IDA)$. This step fails because the $SHA1$ function is collision resistant, so $SHA1(m \mid t) \neq SHA1(m \mid t\prime)$. The attacker must therefore send $m\prime = m$ and $s\prime = \sigma \mid t \mid ID_h$ for the verification to succeed. In this case, the first test $t_r - t \leq t_h$ fails, and the message is rejected. It is therefore impossible for an attacker to replay a message m more than t_h after it has been sent. The key hijacking attack scenario is similar to the directory attack. Three legitimate users, A, B, and E possess the identities and public keys ID_i and p_i for $i \in A, B, E$, all three navigate within the transmission range of each other. Ship A sends a position report m accompanied by its signature $s = \sigma \mid t \mid ID_m$. Vessel B does not have A's public key and sends a public key request as specified in the protocol. When A responds to the request from p_A, E scrambles the message using a powerful antenna and an *SDR* and then immediately sends p_E. B expects to receive the key from A and modify its directory by associating the key p_E with the MMSI ID_A. Then, E created a false message $m\prime$ having as MMSI ID_A and its signature $s\prime = \sigma\prime \mid t\prime \mid ID_{m\prime}$, with $\sigma\prime = sign(SHA1(m\prime \mid t), p_E)$. Upon receipt of these messages, B retrieves the key p_E, which he thinks is A's key, without his directory. In the same way, as in the attack against the directory, the verification of the signature has failed because the identity used to generate the key for E is not the same as the identity used for verification. An attacker cannot, therefore, impersonate another user.

6 IMPLEMENTATION AND EXPERIMENTS

This section details the proof of concept built to demonstrate the feasibility of the solution. The first part presents the material structure of the proof of concept, and the second part details the choices made regarding the code. Finally, the last part details the results obtained thanks to this proof of concept.

6.1 Hardware

The messages are verified using a BeagleBone Black rev3 board running on Linux Debian 11. This board acts as an intermediary between the AIS receiver and the computer displaying the received AIS messages. This computer runs the ECIDS software TimeZero to display the AIS information. The transmission of the messages is done thanks to a HackerOne SDR. The SDR is connected to a laptop computer running Linux. The communications between the LO, the IOs, and the users have been simulated thanks to a web server running on Linux. Figure 4 schematizes the architecture of the proof of concept.

Since the AIS receiver cannot transmit signals, the communications are unidirectional. Therefore, it is not possible to test the public key request. Using an AIS transponder could have allowed an exchange with the SDR, but such transponders emit very strong radio signals. Therefore, a much heavier infrastructure, such as a faraday cage, would have been required to prevent false signals from being emitted outside the

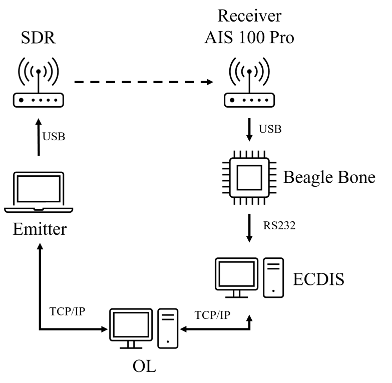


Figure 4: Proof of concept architecture.

lab room. Since the SDR has a minimum transmission power that does not carry the signals more than a few meters, the problem does not arise in the current architecture. In addition, a transponder is significantly more expensive than an AIS receiver.

6.2 Implementation

6.2.1 Choice of Architecture

Because IOs' role in the protocol is purely administrative, they are ignored in this proof of concept. To accommodate this change, the registration phase (phase 1) is ignored. Users send their registration requests directly to the LO. Two-factor authentication is a security measure that is not simulated in the proof of concept, as it is not relevant to prove the feasibility of the protocol. Communications between the LO and users are simply done through HTTP requests. Although no free implementation of the signature protocol is available, it is possible to build one using the PBC library. This library implements the mathematical elements necessary for the functioning of the chosen CLS protocol. The Python version of this library, PyPBC, is used to create two classes, which include the following methods:

- **KGC Class.** The KGC class implements the functions *Initialization* that generate public parameters and KGC master keys and *Registration* that generates a partial key from an identity.
- **User Class.** This class contains functions *GenerateKeys* for generating user keys from partial keys, *Sign* for signing messages, and *Verify* for verifying signatures.

The functions are implemented following the protocol described by (Du et al., 2020).

6.2.2 Initialization Phase

During the initialization phase, the LO runs the functions *GeneratePublicParam* and *MasterKeyGen*. Then, it launches a web server allowing communication with the users. This web server supports the following requests:

- **GET/params.** Returns the public parameters of the KGC.
- **GET /public-keys.** Returns a dictionary containing the public keys of registered users, associated with their MMSI.
- **GET /revocation.** Returns a list of revoked public keys.
- **POST /register.** Accepts an MMSI, and returns a partial key calculated using *PartialKeygen*.
- **POST /send_pk.** Accepts a user's public key and its MMSI, to update the public key directory.

6.2.3 Operational Phase

At startup, the device goes through its initialization phase. If the user is not yet registered, it sends a request to the KGC to register. Upon receiving the partial key, the device generates its user key pair and sends its public key to the LO to update the directory. Then, the device makes two requests to the KGC to retrieve the public and revocation key directories. If the device can issue messages, it initiates the message signing procedure detailed in Algorithm 1. The *sign_every* parameter designates the number of non-critical messages that will not be signed after sending a signed non-critical message. The number of unsigned non-critical messages sent after a signed non-critical message is stored in the variable *unsigned*.

Algorithm 1: Message Signing.

```
Require: :  Sign_every; //  The number of non-critical messages that will
  not be signed after sending a signed non-critical message/

Require: :  unsigned ← ∞; // The number of unsigned non-critical messages
  sent after a signed non-critical message
  for $m \in$ messages do
      $send(m)$;
      if (is_critical($m$) or unsigned >= sign_every) then
          $t \leftarrow timestamp()$;
          $\sigma \leftarrow sign(SHA1(m|t))$;
          $ID_m \leftarrow SHA1(m)[0:4]$;
          $send(\sigma|t|ID_m)$;
          if not (is_critical($m$)) then
              unsigned ← 0;
          end if
      else if not (is_critical($m$) and unsigned < sign_every) then
          unsigned ← unsigned+1;
      end if
  end for
```

If the device can receive messages, the message verification procedure is described in Algorithm 2. The three parameters of the algorithm are a dictionary of user public keys and a list of revoked keys, initialized during the initialization phase, and a boolean describing the transmission capability of the device.

Algorithm 2: Message verification.

```
Require: : $pk : mmsi \rightarrow public\_key$
Require: : $revocated\_keys$ // Revoked key list
Require: : $can\_emit : bool$ // Describe the device's transmission capability
  $buffer \leftarrow Dict()$; // Dictionary of user public keys
  for $m \in messages\_received$ do
      $sender \leftarrow m.mmsi$;
      if not (is_critical($m$) and is_authenticate($sender$)) then
          $accept(m)$;
      else if (is_signature($m$)) then
          $\sigma, t, ID_m \leftarrow decompose(m)$;
          if $((sender, ID_m) \notin buffer)$ then
              continue;
          end if
          $m, t_r =$ buffer.$pop(sender, ID_m)$;
          if not $(t <= t_r$ and $t_r - t < t_l)$ then
              $reject(m)$;
          end if
          if not (sender $\in pk$ and can_emit) then
              $ask\_public\_key(sender)$;
          end if
          if $verify(SHA1(m|t), \sigma, pk[sender], sender)$ then
              $accept(m)$;
              if not (is_critical($m$)) then
                  $authenticate(sender)$;
              end if
          else
              $reject(m)$;
          end if
      else if (is_public_key($m$)) then
          $update\_public\_keys(sender, m)$;
      else if (is_pk_request($m$) and $can\_emit$) then
          $send\_public\_key()$;
      else
          $ID_m = SHA1(m)[0:4]$;
          $buffer[(sender, ID_m)] = (m, timestamp())$;
      end if
  end for
```

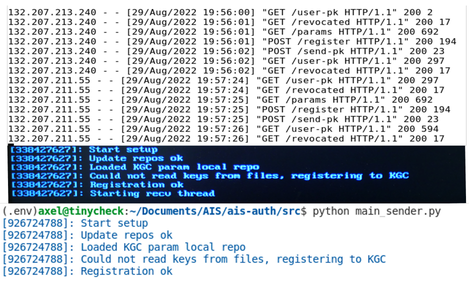
Accepted messages are transmitted to TimeZero without modification. To improve visibility, messages that are rejected, or that were received more than 20 seconds ago without a signature, are transmitted to TimeZero with a type 5 message containing "[UNSIGNED]" in place of the vessel name. Thus, these messages will be visible on TimeZero, but the AIS target will be designated as unsigned.

6.3 Experiments

The goals of the experiments are to verify that the implemented system is functional, to evaluate the experimental network overload to compare it to the theoretical overload and to implement a key hijacking attack to experimentally confirm the security of the protocol against this attack.

6.3.1 Setup

To begin, the LO script is run on a remote server, followed by client scripts executed on the BeagleBone board in receive-only mode and on the computer connected to the SDR in transmit mode. We can see in Figure 5 from top to bottom, the output of the LO and client scripts. On the screen of the LO at the top, we see the initialization phase, which corresponds to the generation of the cryptographic parameters. The following lines are the requests made by the users to register. On the middle and bottom screens, we see the clients making registration requests to the LO, generating their keys, and sending their public keys to the LO. Finally, the clients retrieve the two directories provided by the LO.

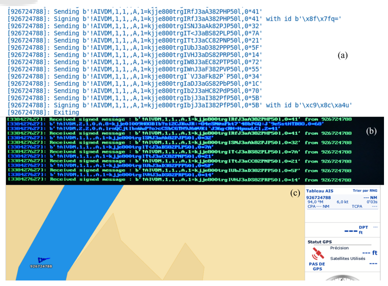


```
132.207.213.240 - - [29/Aug/2022 19:56:00] "GET /user-pk HTTP/1.1" 200 2
132.207.213.240 - - [29/Aug/2022 19:56:01] "GET /revocated HTTP/1.1" 200 17
132.207.213.240 - - [29/Aug/2022 19:56:01] "GET /params HTTP/1.1" 200 692
132.207.213.240 - - [29/Aug/2022 19:56:01] "POST /register HTTP/1.1" 200 194
132.207.213.240 - - [29/Aug/2022 19:56:02] "POST /send-pk HTTP/1.1" 200 23
132.207.213.240 - - [29/Aug/2022 19:56:02] "GET /user-pk HTTP/1.1" 200 297
132.207.213.240 - - [29/Aug/2022 19:56:02] "GET /revocated HTTP/1.1" 200 17
132.207.211.55 - - [29/Aug/2022 19:57:24] "GET /user-pk HTTP/1.1" 200 297
132.207.211.55 - - [29/Aug/2022 19:57:24] "GET /revocated HTTP/1.1" 200 17
132.207.211.55 - - [29/Aug/2022 19:57:25] "GET /params HTTP/1.1" 200 692
132.207.211.55 - - [29/Aug/2022 19:57:25] "POST /register HTTP/1.1" 200 194
132.207.211.55 - - [29/Aug/2022 19:57:25] "POST /send-pk HTTP/1.1" 200 23
132.207.211.55 - - [29/Aug/2022 19:57:26] "GET /user-pk HTTP/1.1" 200 594
132.207.211.55 - - [29/Aug/2022 19:57:26] "GET /revocated HTTP/1.1" 200 17
[338427627]: Start setup
[338427627]: Update repos ok
[338427627]: Loaded KGC param local repo
[338427627]: Could not read keys from files, registering to KGC
[338427627]: Registration ok
[338427627]: Starting recv thread
(.env)axel@tinycheck:~/Documents/AIS/ais-auth/src$ python main_sender.py
[926724788]: Start setup
[926724788]: Update repos ok
[926724788]: Loaded KGC param local repo
[926724788]: Could not read keys from files, registering to KGC
[926724788]: Registration ok
```

Figure 5: Registration of the two users from the point of view of the LO (top), the receiver (middle), and the transmitter (bottom).

6.3.2 Solution Validation

The purpose of the first experiment is to ensure that the system is operational. The client script on the computer connected to the SDR transmits a series of AIS messages corresponding to position reports (type 1 messages) generated by NEMA Studio, a software program that, among other things, generates AIS messages. Figure 6 shows the screens of the transmitting computer (a), the BeagleBone chart (b), and the ECDIS (c). On the screen of the transmitter, the messages sent can be seen, the signed messages are preceded by the mention of "signing". On the BeagleBone screen, accepted messages are displayed in green. Finally, the AIS target is correctly displayed on the ECDIS.

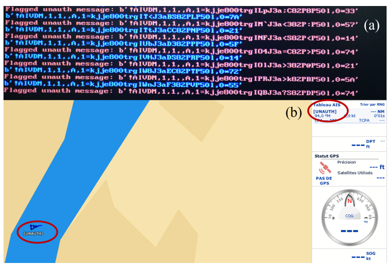


```
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIRfJ3aA382PHP50l,0*41'
[926724788]: Signing b'!AIVDM,1,1,,A,1=kjje800trgIRfJ3aA382PHP50l,0*41' with id b'\x8f\x7fq='
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgISNJ3aAk82PJP50l,0*32'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIT<J3aBS82PLP50l,0*7A'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgITtJ3aCC82PNP50l,0*21'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIUbJ3aD382PPP50l,0*5F'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIVHJ3aDS82PRP50l,0*14'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIW8J3aEC82PTP50l,0*72'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIWnJ3aF382PVP50l,0*55'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgI`VJ3aFk82P`P50l,0*34'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIaDJ3aGS82PbP50l,0*1C'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIb2J3aHC82PdP50l,0*70'
[926724788]: Sending b'!AIVDM,1,1,,A,1=kjje800trgIbjJ3aI382PfP50l,0*5B'
[926724788]: Signing b'!AIVDM,1,1,,A,1=kjje800trgIbjJ3aI382PfP50l,0*5B' with id b'\xc9\x8c\xa4u'
[926724788]: Exiting
```

```
[338427627]: Received signed message : b'!AIVDM,1,1,,A,1=kjje800trgIRfJ3aA382PHP50l,0*41' from 926724788
[338427627]: b'!AIVDM,2,1,0,A,8=kjje0I009H8OBIb?si2CJKvJR:=O4eSN@nFkt7`48bPGQ:J`9e5stHTB8O,0*68'
[338427627]: b'!AIVDM,2,2,0,A,lrnQCjtIbnWwP?o>cCSbCGTH9J6@VKi`J36g<RH=HpuwLCl,2*41'
[338427627]: Received signed message : b'!AIVDM,1,1,,A,1=kjje800trgIRfJ3aA382PHP50l,0*41' from 926724788
[338427627]: b'!AIVDM,1,1,,A,1=kjje800trgISNJ3aAk82PJP50l,0*32'
[338427627]: Received signed message : b'!AIVDM,1,1,,A,1=kjje800trgISNJ3aAk82PJP50l,0*32' from 926724788
[338427627]: b'!AIVDM,1,1,,A,1=kjje800trgIT<J3aBS82PLP50l,0*7A'
[338427627]: Received signed message : b'!AIVDM,1,1,,A,1=kjje800trgIT<J3aBS82PLP50l,0*7A' from 926724788
[338427627]: b'!AIVDM,1,1,,A,1=kjje800trgITtJ3aCC82PNP50l,0*21'
[338427627]: Received signed message : b'!AIVDM,1,1,,A,1=kjje800trgITtJ3aCC82PNP50l,0*21' from 926724788
[338427627]: b'!AIVDM,1,1,,A,1=kjje800trgIUbJ3aD382PPP50l,0*5F'
[338427627]: Received signed message : b'!AIVDM,1,1,,A,1=kjje800trgIUbJ3aD382PPP50l,0*5F' from 926724788
[338427627]: b'!AIVDM,1,1,,A,1=kjje800trgIVHJ3aDS82PRP50l,0*14'
[338427627]: Received signed message : b'!AIVDM,1,1,,A,1=kjje800trgIVHJ3aDS82PRP50l,0*14' from 926724788
```

Figure 6: Sending of signed position reports (a), verification and acceptance of messages (b) and corresponding AIS target on ECDIS (c).

The experiment is then repeated, but an error is introduced in the signatures. For this, a line of code is added after the calculation of the signature to change its last bit. We can see in Figure 7 that this time, the messages are rejected by the BeagleBone and are displayed in red (a). On the screen of the ECDIS (b), the target is displayed, but with the annotation ”[UNAUTH]”, indicating that the messages were rejected.



```
Flagged unauth message: b'!AIVDM,1,1,,A,1=kjje800trgILpJ3a;C82P8P501,0*33'
b'!AIVDM,1,1,,A,1=kjje800trgIT<J3aBS82PLP501,0*7A'
Flagged unauth message: b'!AIVDM,1,1,,A,1=kjje800trgIM`J3a<382P:P501,0*57'
b'!AIVDM,1,1,,A,1=kjje800trgITtJ3aCC82PNP501,0*21'
Flagged unauth message: b'!AIVDM,1,1,,A,1=kjje800trgINFJ3a<S82P<P501,0*14'
b'!AIVDM,1,1,,A,1=kjje800trgIUbJ3aD382PPP501,0*5F'
Flagged unauth message: b'!AIVDM,1,1,,A,1=kjje800trgIO4J3a=C82P>P501,0*74'
b'!AIVDM,1,1,,A,1=kjje800trgIVHJ3aDS82PRP501,0*14'
Flagged unauth message: b'!AIVDM,1,1,,A,1=kjje800trgIOlJ3a>382P@P501,0*21'
b'!AIVDM,1,1,,A,1=kjje800trgIW8J3aEC82PTP501,0*72'
Flagged unauth message: b'!AIVDM,1,1,,A,1=kjje800trgIPRJ3a>k82PBP501,0*5A'
b'!AIVDM,1,1,,A,1=kjje800trgIWnJ3aF382PVP501,0*55'
Flagged unauth message: b'!AIVDM,1,1,,A,1=kjje800trgIQBJ3a?S82PDP501,0*74'
```

Figure 7: Signature verification fails (a), the AIS target is tagged with ”UNAUTH]” (red circles) on the ECDIS (b).

6.3.3 Overhead Estimation

The second experiment aims at measuring the AIS network overload induced by the solution, with different values of the safety parameter S_e defined in Equation 3. These measurements will allow us to validate the theoretical results, and to estimate values of S_e that can be used in practice. To do this, we use a sample of AIS messages from the *aishub.net* platform. This sample consists of more than 85,000 AIS messages from all over the world, sent over a period of 3 minutes. To measure the effect of introducing signatures, Algorithm 3 is used to simulate the operation of the protocol. Each message in the file is analyzed, and a counter of radio slots is incremented according to whether or not a signature is introduced.

Algorithm 3: Overhead estimation.

```
$auth \leftarrow HashMap()$;
$sent \leftarrow 0$;
for ($m \in messages$) do
    $mmsi \leftarrow m.mmsi$;
    if ($is\_critical(m)$ or $auth[mmsi] \geq S_e$) then
        $sent \leftarrow sent + 3$;
        if not ($is\_critical(m)$) then
            $auth[mmsi] \leftarrow 0$;
        end if
    else if not ($is\_critical(m)$) then
        $auth[mmsi] \leftarrow auth[mmsi] + 1$;
    end if
    $sent \leftarrow sent + len(m)$;
end for
```

The algorithm is run for several values of the security parameter S_e. The results are given in the table 4. The number of emission slots after adding the signatures is computed for each value of S_e, allowing a comparison of the experimental overload and the theoretical overload given by Equation 6

$$overload = 3(1 - 0.9735(1 - \frac{1}{S_e})) \qquad (6)$$

The significant differences between the theoretical and experimental values can be explained by the data sample used. The data is recorded over a 3-minute window, but many non-critical messages are broadcast over one minute or more. The sample used therefore induces an overrepresentation of signed non-critical messages because these are the first non-critical messages sent by a sender. The theoretical overload is an average overload, reached asymptotically. To solve this problem, we can repeat the sample so that the overload values reach their asymptotic values. Repeating the sample makes the data content inconsistent (ships will jump back and forth, for example), but this content is not analyzed in the experiment. The features that are useful to the experiment, i.e., message transmission frequencies, proportions of different message types, and MMSIs associated with the messages, are retained. This solution could introduce biases since the measured values will only be those corresponding to a three-minute sample, and there is no guarantee that the communications do not evolve after these three minutes. In addition, the repeated sample does not model vessels leaving and entering the AIS network. To determine how many repetitions of the sample are needed to measure the overloads, we can increase them gradually until the value stabilizes. Figure 8 shows that at 10 or more repetitions, the overload is stabilized. An identical analysis for other values of S_e shows similar results. We choose 15 repetitions, to ensure that the measured values are properly stabilized.

Table 4: Experimental overload and theoretical overload as a function of the safety parameter S_e.

S_e	Experimental overload	Theoretical overload
5	75.8%	66.3%
10	56.7%	37.2%
15	51.8%	27.4%
20	47.3%	22.6%

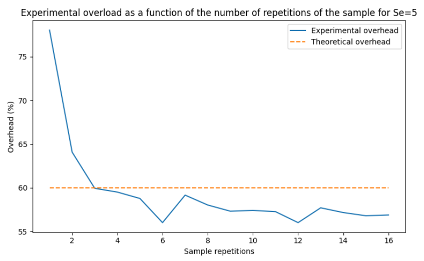


Figure 8: Experimental overload as a function of the number of repetitions of the sample for $S_e = 5$.

Finally, Table 5 shows that the experimental results do not deviate by more than 10.3% from the theoretical values and that the experimental overloads are always smaller than the theoretical values for the S_e used in the experiment.

Table 5: Experimental and theoretical overload as a function of the safety parameter S_e, with 12 sample repetitions.

S_e	Experimental overload	Theoretical overload
5	56.0%	66.3%
10	35.9%	37.2%
15	26.7%	27.4%
20	22.7%	22.6%

This experimentation allows us to conclude that the theoretical values of the overloads are quite accurate and can be used to decide the value of S_e to use according to the situation.

6.3.4 Key Hijacking Simulation

The objective of the third experiment is to verify the security of the protocol against a key hijacking attack. We have chosen this attack because it allows us to show the counter-intuitive properties of uncertificated cryptography, which allows us to mathematically link a key and an identity. The scenario of the attack is as follows. Three legitimate ships A, B, and E, all registered with the KGC, are sailing in the same area. Ship A does not have B's public key. B sends a position report message m_1, with its signature $s_1 = \sigma_1|t_1|ID_1$. Ship B does not have A's public key, so it sends a public key request to A. Equipped with a powerful SDR, E scrambles A's response and sends its public key p_E instead. The ship E then sends the message m_2, using A's MMSI, but signing with its private key. To simulate this situation, three clients A, B, and E, with respective MMSIs of 111111111, 222222222, and 333333333 are created and registered with the LO. The B user key directory is manually modified to contain the public key of E instead of that of A. This method places the devices in the same state as after a key hijacking described in the previous paragraph. Then, E sends a message containing A's MMSI that he signs with his private key. In Figure inc_FIGURE, we can see that the message is rejected by B, despite its directory being attacked by E. The protocol is resistant to this attack.

Finally, we analyze the limitations of the work that has been done and suggest areas for improvement for future work. One of the AIS experiments did not allow a definitive conclusion to be drawn due to a lack of data. One possible solution is data simulation. This task is complex because it is necessary to ensure the quality of the simulated data so as not to introduce bias into the experiments. The acceptance of the authentication protocol requires further experiments, such as deploying a proof of concept and studying compatibility with current devices. AIS is a critical maritime security asset, but ship computer systems are complex and use many protocols. It would be interesting to study the security of industrial protocols used by ship control systems.

7 CONCLUSION

In this work, we have built and implemented a protocol that overcomes the biggest vulnerability of AIS: the lack of authentication. We first identified the design constraints imposed by AIS, then we selected technologies and an architecture based on these constraints. An estimation of the network overhead allowed us to build a protocol that does not endanger the stability of the AIS network. Finally, we implemented this protocol on a microprocessor and carried out a proper functioning test, an experimental estimation of the overload induced by the introduction of signatures, and the simulation of an attack to experimentally show the security of the solution. The proof of concept we developed showed that the solution is realistic and works as expected.

ACKNOWLEDGMENT

We would like to thank our partners Davie and Neptune Cyber for their financial support. Without their generous contributions, this project would not have been possible. We would like to extend our heartfelt thanks to Dr. Pierre Langlois and Jean Yves Ouattara, who shared their time and insights with us.

REFERENCES

Balduzzi, M., Pasta, A., and Wilhoit, K. (2014). A security evaluation of ais automated identification system. In *Proceedings of the 30th annual computer security applications conference*, pages 436–445.

Du, H., Wen, Q., Zhang, S., and Gao, M. (2020). A new provably secure certificateless signature scheme for internet of things. *Ad Hoc Networks*, 100:102074.

Goudosis, A. and Katsikas, S. (2020). Secure ais with identity-based authentication and encryption. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 14(2):287–298.

Goudossis, A. and Katsikas, S. K. (2019). Towards a secure automatic identification system (ais). *Journal of Marine Science and Technology*, 24:410–423.

Hall, J., Lee, J., Benin, J., Armstrong, C., and Owen, H. (2015). Ieee 1609 influenced automatic identification system (ais). In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE.

Last, P., Bahlke, C., Hering-Bertram, M., and Linsen, L. (2014a). Comprehensive analysis of automatic identification system (ais) data in regard to vessel movement prediction. *The Journal of Navigation*, 67(5):791–809.

Last, P., Bahlke, C., Hering-Bertram, M., and Linsen, L. (2014b). Comprehensive analysis of automatic identification system (ais) data in regard to vessel movement prediction. *The Journal of Navigation*, 67(5):791–809.

Merchant, N. D., Witt, M. J., Blondel, P., Godley, B. J., and Smith, G. H. (2012). Assessing sound exposure from shipping in coastal waters using a single hydrophone and automatic identification system (ais) data. *Marine pollution bulletin*, 64(7):1320–1329.

Nguyen, D. H. (2020). Hardening automatic identification systems: Providing integrity through an application of lightweight cryptograph techniques. Technical report, Naval Postgraduate School.

Schwehr, K. D. and McGillivary, P. A. (2007). Marine ship automatic identification system (ais) for enhanced coastal security capabilities: an oil spill tracking application. In *OCEANS 2007*, pages 1–9. IEEE.

Series, M. (2014). Technical characteristics for an automatic identification system using time-division multiple access in the vhf maritime mobile band. *Recommendation ITU: Geneva, Switzerland*, pages 1371–1375.

Struck, M. C. and Stoppe, J. (2021). A backwards compatible approach to authenticate automatic identification system messages. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 524–529. IEEE.

Su, P., Sun, N., Zhu, L., Li, Y., Bi, R., Li, M., and Zhang, Z. (2017). A privacy-preserving and vessel authentication scheme using automatic identification system. In *Proceedings of the Fifth ACM International Workshop on Security in Cloud Computing*, pages 83–90.

Wimpenny, G., Šafář, J., Grant, A., and Bransby, M. (2022). Securing the automatic identification system (ais): Using public key cryptography to prevent spoofing whilst retaining backwards compatibility. *The Journal of Navigation*, 75(2):333–345.

Xiong, H., Qin, Z., and Vasilakos, A. V. (2016). *Introduction to certificateless cryptography*. CRC Press.