Journal of Intelligent & Robotic Systems (2021) 103:26
https://doi.org/10.1007/s10846-021-01472-5

REGULAR PAPER

Secure and Efficient Transmission of Vision-Based Feedback Control Signals

Øystein Volden[1] · Petter Solnør[1] · Slobodan Petrovic[2] · Thor I. Fossen[1]

Received: 17 September 2020 / Accepted: 3 August 2021

Abstract

An ever-increasing number of autonomous vehicles use bandwidth-greedy sensors such as cameras and LiDARs to sense and act to the world around us. Unfortunately, signal transmission in vehicles is vulnerable to passive and active cyber-physical attacks that may result in loss of intellectual property, or worse yet, the loss of control of a vehicle, potentially causing great harm. Therefore, it is important to investigate efficient cryptographic methods to secure signal transmission in such vehicles against outside threats. This study is motivated by the observation that previous publications have suggested legacy algorithms, which are either inefficient or insecure for vision-based signals. We show how stream ciphers and authenticated encryption can be applied to transfer sensor data securely and efficiently between computing devices suitable for distributed guidance, navigation, and control systems. We provide an efficient and flexible pipeline of cryptographic operations on image and point cloud data in the Robot Operating System (ROS). We also demonstrate how image data can be compressed to reduce the amount of data to be encrypted, transmitted, and decrypted. Experiments on embedded computers verify that modern software cryptographic algorithms perform very well on large sensor data. Hence, the introduction of such algorithms should enhance security without significantly compromising the overall performance.

Keywords Cyber-security · Authentication · Encryption · Compression · Sensors · ROS · Autonomous vehicles

1 Introduction

Autonomy has gained increased traction in the maritime sector over the past decade. By promising reduced costs and improved safety, unmanned surface vehicles (USVs), autonomous underwater vehicles (AUVs), and unmanned aerial vehicles (UAVs) may revolutionize services such as data acquisition, mapping, and remote surveillance [1]. However, significant challenges remain before autonomous vehicles are ready to enter the commercial market. In particular, these vehicles must be secure and robust against the growing threat of cyber-physical attacks for wide-spread acceptance by authorities, classification societies, and the general public [2]. In this context, we investigate how we can use cryptographic algorithms to protect sensor data in time-critical applications. The goal is to secure autonomous vehicles against passive and active adversaries with access to the transmission lines, as shown in Fig. 1.

Autonomous vehicles are controlled by guidance, navigation, and control (GNC) systems that perform path planning, estimate position, velocities, and attitude, and compute appropriate control signals to execute, respectively [3]. Often, these systems are modular since each task is performed by separate computational devices. The communication between these components is often done through local area networks, to which sensor devices are also connected. Feedback control systems that close the loop through networks are commonly referred to as

✉ Øystein Volden
oystein.volden@ntnu.no

Petter Solnør
petter.solnor@ntnu.no

Slobodan Petrovic
slobodan.petrovic@ntnu.no

Thor I. Fossen
thor.fossen@ntnu.no

[1] Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway

[2] Department of Information Security and Communication Technology, Norwegian University of Science and Technology, 2802 Gjøvik, Norway

Published online: 09 September 2021

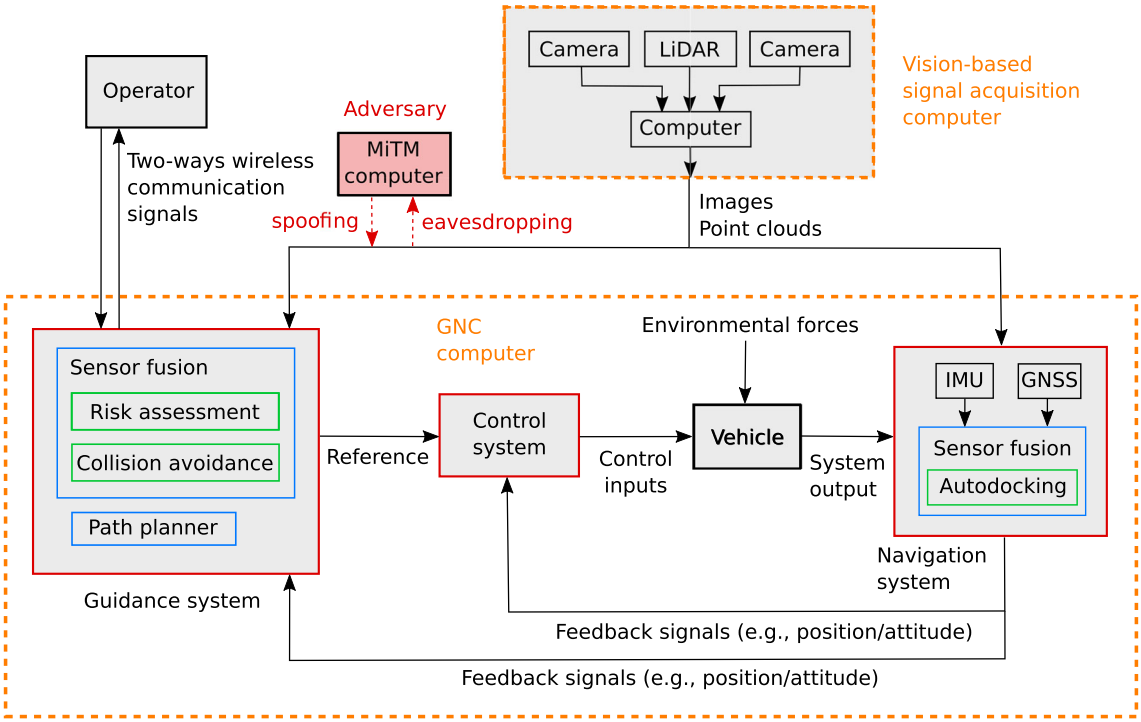


Fig. 1 The figure shows a guidance, navigation, and control (GNC) system under attack by an adversary

networked control systems (NCSs) [4]. By providing ease of installation, reduced maintenance costs, and flexible architectures, NCSs show promising advantages over systems with independent, dedicated communication channels [5].

For safety-critical tasks such as collision avoidance and autonomous docking, the GNC system must produce a detailed overview of the vehicle's environment. This is typically achieved with vision-based sensors such as cameras and LiDARs [6–8]. Because collision avoidance and autonomous docking are independent tasks performed by the guidance and navigation systems, respectively, they may require access to the same measurements. By using a dedicated computer for synchronized data acquisition, the raw data may be securely transmitted to the required systems, as seen in Fig. 1. Using local feature extraction to reduce the amount of data transmitted from the vision-based signal acquisition computer is possible. However, such a solution would blur the borders of the modular GNC design and increase system complexity. Alternatively, the data could be compressed before transmission, but compression may reduce the data quality and induce additional latency. It is therefore important to understand how compression algorithms affect the overall system performance.

The transmission of these signals, however, present attack surfaces which adversaries may exploit. Because of real-time requirements, the User Datagram Protocol (UDP) over the Internet Protocol (IP) is a common solution for signal transmission of large sensor data due to the simplicity of the protocols and the high bandwidth available [9]. Unfortunately, these protocols are inherently insecure and vulnerable to a number of cyber-physical attacks such as eavesdropping, bit manipulation, and packet injection by adversaries. If an adversary gains access to the signal transmissions in autonomous vehicles, he/she is free to eavesdrop on the data and gain access to confidential system information. Furthermore, the adversary could inject its own data into the signal transmission to manipulate the behavior of the vehicle, as seen in Fig. 1. Such attacks may, for example, result in the failure of collision avoidance systems with dramatic consequences; rather than tracking incoming obstacles, the vehicle may be fooled into 'avoiding' objects that do not exist.

Traditionally, cryptography has been used to solve such problems. To prevent unauthorized parties from accessing the transmitted data, the data streams should be *encrypted*. To prevent the injection of spoofed data, the origin of the data streams should be *authenticated*. However, the large throughput required when processing images and point clouds may result in large time delays. This paper aims to resolve this problem by demonstrating how state-of-the-art cryptographic algorithms result in considerably smaller time delays than existing works have suggested. We also seek to investigate whether compression before encryption can accelerate the cryptographic operations while also reducing the required throughput.

1.1 Related Work

Since NCSs connect system components across a network, they become vulnerable to cyber-physical attacks such as eavesdropping and deception attacks, as described in [10, 11]. Therefore, the use of cryptographic algorithms such as the Data Encryption Standard (DES) [12], Triple DES (3DES) [13], Advanced Encryption Standard (AES) [14], Message Digest 5 (MD5) [15], and Keyed-Hash Message Authentication Code (HMAC) [16] has been suggested by [17–19] to secure the signal transmission in NCSs. In particular, Pang & Liu [18] suggested a 'secure transmission mechanism' consisting of a block cipher and a cryptographic hash function in a 'hash-then-encrypt' composition to provide security against deception attacks. However, the use of ad-hoc schemes to prevent deception attacks, such as the scheme proposed by Pang & Liu, has been shown to be cryptographically weak [20]. Instead, we argue that proper authenticated encryption, obtained through dedicated authenticated encryption algorithms or cryptographically strong compositions, e.g., those investigated by Bellare & Namprempre [21], should be used.

Case-studies examining implementation-specific software such as the Robot Operating System (ROS), for example, by Teixeira et al. [22], have discovered similar weaknesses to eavesdropping and data manipulation attacks. To counteract such attacks, well-known cryptographic algorithms such as 3DES, Blowfish [23], and AES have been applied in ROS [24–26]. Specifically, Rodrigues-Lera et al. [25] investigated the use of 3DES, AES, and Blowfish on images and LiDAR data in the ROS environment and found system performance to be adversely affected by the cryptographic operations. Due to the computational overhead, cryptographic algorithms induce latencies that are important to consider, especially if real-time requirements apply. Since the latency induced by cryptographic operations grows linearly with the data size, these latencies may be significant for large data such as images and point clouds.

The work described above examined the use of block ciphers accessed through open-source libraries. This leads Teixeira et al. to conclude that cryptography is not viable in all cases because the latency induced by encryption and decryption compromises real-time performance [22]. Interestingly, the use of state-of-the-art stream ciphers, such as those found in the eSTREAM portfolio [27] and the AEGIS cipher [28], has not been considered by any of the previous work. Stream ciphers are stateful ciphers that usually consist of an *initialization phase* and a *keystream generation phase*. While the initialization phase does result in an initial overhead compared to block ciphers, the keystream generation phase of a stream cipher is much more efficient than that of a block cipher. This is important because vision-based signals are large compared to more common data transmitted in feedback control systems. For example, the transmission of a state estimate containing position, velocities, and attitude at a rate of 100 Hz would require a bandwidth around 10 KB/s. In comparison, images and point-clouds transmitted at 10 Hz would require bandwidth at least three orders of magnitude greater. Therefore, for encryption of vision-based signals, we suggest the use of dedicated stream ciphers, for which we expect to produce considerably better results in terms of latency.

1.2 Main Contributions

The main objective of this study is to identify efficient cryptographic algorithms that avoid critical time delays in the feedback loop for autonomous vehicles. This is particularly important for vision-based signals processed by embedded systems onboard vehicles where the computational power is limited. To this end, we demonstrate that a stream of images and point clouds can be transmitted securely and efficiently between embedded systems by using modern cryptographic methods. We show how authenticated encryption can be used to obtain confidentiality, integrity, and data origin authenticity by combining stream ciphers and message authentication codes, or through a dedicated authenticated encryption algorithm. Finally, we demonstrate that by using the proposed algorithms, the use of compression before cryptographic operations results in larger time delays. Therefore, compression should only be applied if the network bandwidth is constrained. By suggesting the use of stream ciphers and authenticated encryption instead of block ciphers, this paper presents an important contribution to the development of secure autonomous vehicles. Experimental results verify that the proposed algorithms significantly outperform algorithms suggested by others and should be used in autonomous vehicles.

The software-oriented stream ciphers from the eSTREAM portfolio are assessed and compared against the de facto standard, i.e., the AES algorithm. The best-performing encryption algorithms are then composed with the HMAC algorithm to obtain authenticated encryption and then compared with the AEGIS algorithm. Finally, we investigate whether the use of compression before cryptographic operations results in increased performance. The algorithms have been integrated into the ROS environment as a cryptographically strong pipeline that enables cryptographic operations on image and point cloud data. The proposed pipeline is flexible since the data type is irrelevant to the implementation. Efficiency is ensured since the computational complexity is inherited from the underlying cryptographic algorithms and grows linearly with the data size. Source code, data set, and instructions to run the algorithms in ROS have been made available in a public

Github repository [29]. The dataset is based on data collection onboard a USV and was created by the authors. To assess algorithm performance, experiments were conducted on edge-computing embedded devices rather than high-performance desktop computers. Although the experiments are demonstrated for USVs, the embedded systems in use apply to a wide range of autonomous vehicles.

1.3 Outline

This paper is structured as follows. Section 2 introduces the implemented methods and discusses how they are realized in ROS. Section 3 describes the hardware, software, and sensor data used in the experiments, the laboratory setup, and experiment-specific details. It also includes results and discussions regarding the experiments. Finally, we summarize the most important findings and how they relate to the results achieved by other authors in Section 4.

2 Algorithms and Implementation

First, we introduce the proposed cryptographic pipeline, followed by an introduction to ROS. Then, an overview of the relevant algorithms is given. Finally, we describe how the algorithms are integrated into the ROS environment. The focus lies on the communication between the vision-based signal acquisition computer and the guidance and navigation computers but applies to all signal transmissions seen in Fig. 1.

Introducing our analogy, used throughout this paper, one of the computers acts as the *talker* while the other computer acts as the *listener*. The talker is the node that encrypts and transmits the sensor signals, while the listener is the node that receives and recovers the original message through decryption, as seen in Fig. 2. Authentication is included in the pipeline if the encryption algorithm does not provide data origin authenticity directly. That is, the talker node computes a tag based on the message before the message and tag are transmitted to the listener. Upon reception, the listener recomputes the tag and compares it with the received tag to validate the message's authenticity.

2.1 Communication and Sensor Interfacing in ROS

ROS is a flexible open-source framework for robot software development and is designed with distributed computing in mind. An essential part of ROS concerns how different computational processes can communicate and interchange messages. In this context, ROS *nodes* play an important role. In general, nodes are meant to operate at a fine-grained scale. Hence, robot control

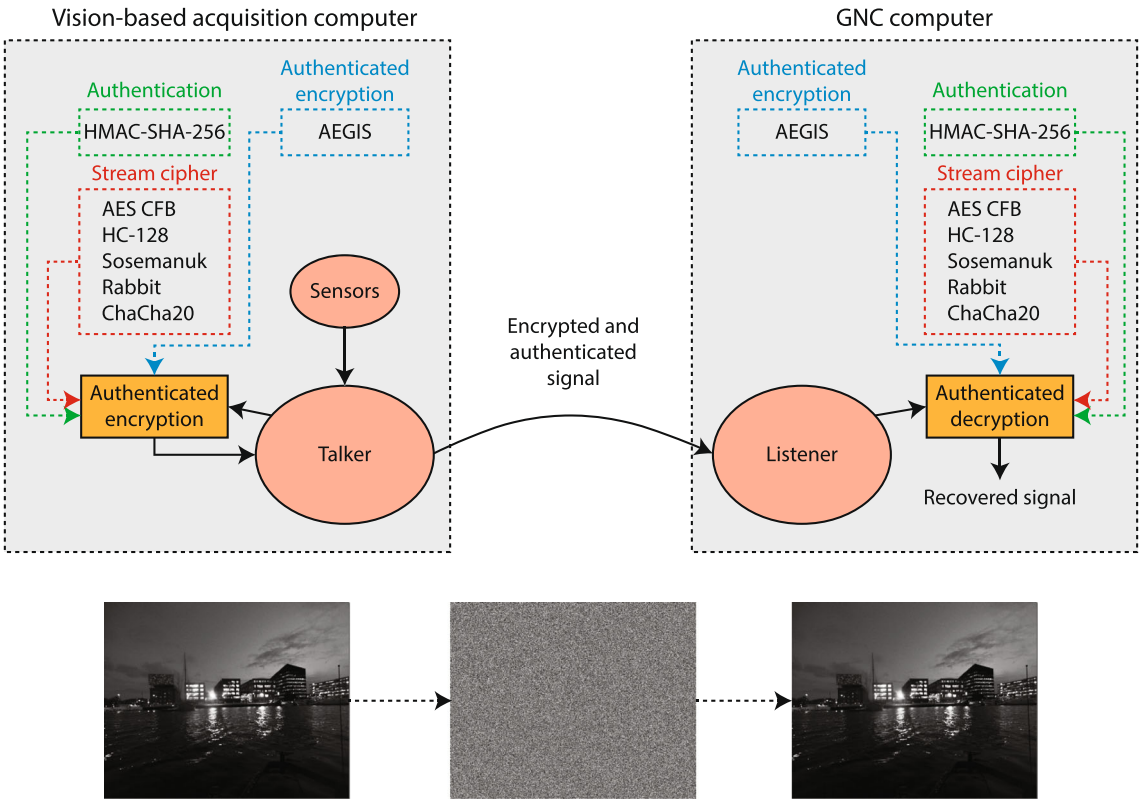


Fig. 2 The block diagram shows the proposed cryptographic pipeline implemented in ROS

systems usually comprise several nodes. The use of multiple nodes also provides several benefits, such as reduced code complexity and additional fault tolerance as crashes are isolated to individual nodes. ROS *topics* are closely related to ROS nodes. Topics are named buses over which nodes exchange messages. ROS topics are intended for unidirectional streaming communication under the publisher and subscriber scheme. In general, nodes are not aware of to whom they are communicating. Instead, nodes subscribe to the topic containing data of interest generated by other nodes publishing data to the relevant topic. There can be multiple publishers and subscribers to a single topic. Furthermore, ROS supports both Transmission Control Protocol (TCP)/IP-based and UDP/IP-based message transport. The TCP/IP-based transport protocol, known as TCPROS, is the default communication protocol in ROS and streams data over persistent connections. The UDP/IP-based transport, known as UDPROS, is a low-latency, lossy alternative which separates messages into UDP packets. ROS nodes usually negotiate the desired transport at runtime. Nevertheless, it is possible to specify the choice of transport protocol manually.

This paper focuses on large sensor data such as images and point clouds, both categorized under the *sensor_msgs* package in ROS. The cryptographic algorithms require that the data to be processed is *contiguous* in memory. Fortunately, data fields categorized under the *sensor_msgs* package are already serialized. We may therefore manipulate the data fields directly with cryptographic operations. After encryption, the encrypted data is represented as a byte stream which is easily embedded in a ROS topic through the *roscpp* library. The roscpp library enables C++ programmers to quickly interface with ROS topics and is designed to be the high-performance library for ROS.

2.2 Cryptographic Algorithms

The cryptographic algorithms described in this paper are *symmetric* in the sense that *a shared secret key* is used for encryption and decryption, and authentication and validation, respectively. The only *secret* part of the cryptographic algorithms is material directly derived from the secret key, i.e., the algorithms are entirely public. An encryption algorithm is considered *broken* if there exists a *reasonable* attack that is *more efficient* than a *brute-force attack*, i.e., an exhaustive search through the key space. The input to the encryption algorithm is referred to as *plaintext*, while the output is referred to as *ciphertext*. By decrypting the ciphertext, the original plaintext is recovered as it was before encryption was applied.

Table 1 An overview of the cryptographic algorithms, the services they provide, and original work

Algorithm	Encryption	Authentication	Original work
AES	✓		[14]
HC-128	✓		[30]
ChaCha	✓		[31]
Rabbit	✓		[32]
Sosemanuk	✓		[33]
HMAC-SHA-256		✓	[16, 34]
AEGIS	✓	✓	[28]

Symmetric encryption algorithms are divided into two categories: block ciphers and stream ciphers. Block ciphers are stateless substitutions parametrized by a K-bit key operating on B-bit blocks. Examples are DES [12] and AES [14], which are well-known block ciphers. Since encryption of the same plaintext would always result in the same ciphertext, block ciphers are usually operated in specific *modes*, such as the Cipher Block Chaining (CBC) mode and Cipher Feedback (CFB) mode. When a block cipher is operated in the CBC mode it is still considered a block cipher, while the CFB mode converts the block cipher into a *stream cipher* by introducing a state. Stream ciphers work by extending a relatively short secret key into a much longer pseudorandom keystream which is then mixed with the plaintext, usually through the exclusive-or (XOR) operation, to form the ciphertext. Since stream ciphers are stateful, a unique, public parameter known as the *initialization vector* (IV) is mixed with the secret key to produce an initial state of the stream cipher before a message is encrypted or decrypted. In this sense, the IV serves as a cryptographic synchronization mechanism.

An overview of the relevant cryptographic algorithms that were integrated into the ROS environment and assessed can be seen in Table 1. The AES algorithm serves as a benchmark such that the performance of the algorithms may be compared to known results from related works. The algorithm implementations described in [35] were used. Brief descriptions of the algorithms are given. For additional details, the readers are advised to consult the corresponding references given at the end of this paper.

2.2.1 Advanced Encryption Standard

In 1997, the DES algorithm had been in place for over 20 years, and questions regarding the security of DES had haunted DES since its inception. Rather than repeating the closed DES standardization process from the 1970s, the process of finding a new encryption standard, the AES, was decided to be public. Following a 3-year process, with careful examination of all the submissions concerning their

security and performance, the Rijndael block cipher was selected in April 2000 and adopted as a federal information processing standard in early 2001 [14].

AES consists of multiple keyed rounds that operate on 128-bit blocks through a series of substitutions and permutations and quickly became the most popular encryption algorithm in-use. As a result of the widespread adoption of the standard, many processor architectures have implemented enhanced instruction sets, such as the *x86* Intel AES New Instructions (AES-NI) and *ARMv8* Cryptography Extension, that provide hardware acceleration of the AES operations.

2.2.2 eSTREAM Portfolio

During a discussion at the 2004 RSA Data Security Conference, the need for dedicated stream ciphers was questioned following the success of the AES. As an argument *for* the use of dedicated stream cipher designs, Shamir [27, page 1] identified two key areas in which dedicated stream ciphers could offer an advantage over block ciphers, namely: "*(1) where exceptionally high throughput is required in software and (2) where exceptionally low resource consumption is required in hardware*".

As a result of the discussion, the ECRYPT Stream Cipher Project, a multi-year effort that ran from 2004-2008, was launched. The goal was to stimulate work on stream ciphers, and the project resulted in several successful entrants. The collection of successful entrants became known as the eSTREAM portfolio. The ciphers were designed to derive an initial state from a public IV and a secret key and to be optimized for software implementation (Profile 1) or hardware implementation (Profile 2) to address (1) and (2), respectively. Since we integrate the algorithms into the ROS environment, we focus on the stream ciphers optimized for software implementation.

The software-oriented portion of the eSTREAM portfolio consists of the following stream ciphers: HC-128 designed by Wu [30], Rabbit designed by Boesgaard et al. [32], ChaCha20 designed by Bernstein [31], and Sosemanuk designed by Berbain et al. [33]. We note that ChaCha20 is an improved version, with additional security margins and slightly increased performance, of the Salsa20 cipher, which is the original member of the eSTREAM portfolio. While a detailed description of each of the algorithms is out of scope, we mention that the algorithms differ structurally. These differences lead to the belief that if a weakness is identified for one cipher, the other ciphers are likely to remain unaffected. The HC-128 stream cipher uses large permutation tables, and the contents of the permutation tables determine the state. The Rabbit stream cipher uses elements from Chaos Theory, while the ChaCha20 stream cipher is an Add-Rotate-XOR (ARX) cipher. Finally, the Sosemanuk stream cipher uses a composition of a linear feedback shift register associated with a primitive feedback polynomial and parts of the Serpent block cipher, the runner-up submission to the AES competition.

2.2.3 HMAC-SHA-256

Since encryption alone does not provide integrity nor data origin authenticity, a different cryptographic technique must be applied. A message authentication code (MAC) is a symmetric-key construction which takes an arbitrary length input and produces a fixed B-bit output called a *tag*. A MAC algorithm is considered broken if an adversary is capable of forging a valid tag on *at least one message*. This is called an *existential forgery*.

The Keyed-Hash Message Authentication Code (HMAC) [16] is an algorithm that constructs a MAC from an unkeyed cryptographic hash function. In our experiments, we use the Secure Hash Algorithm 256 (SHA-256) [34] as the cryptographic hash function per recommendation from the Internet Engineering Task Force [36]. By composing the HMAC algorithm with the encryption algorithms in the compositions described by [21], authenticated encryption is achieved. In our experiments, we use the 'Encrypt-then-MAC'-composition shown in Fig. 3a.

2.2.4 AEGIS

Instead of using a generic composition such as 'Encrypt-then-MAC', we may use a dedicated authenticated encryption algorithm, as seen in Fig. 3b. The Competition for Authenticated Encryption: Security, Applicability and Robustness (CAESAR) was an effort that ran from 2014-2019. The goal was to identify authenticated encryption schemes that provided better performance than those in use at the time, most notably AES Galois / Counter Mode (GCM) and AES Counter with CBC-MAC (CCM). The AEGIS cipher designed by Bart Preneel and Hongjun Wu was a successful entrant and is part of the high-performance portfolio along with AES Offset Codebook Mode (OCB) designed by Rogaway et al. [37]. Unfortunately, AES OCB is encumbered by patents and is, therefore, not considered here. The AEGIS stream cipher was constructed with the AES round function in mind to take advantage of the AES-NI instruction set, but it also offers strong performance with table-driven variants of the AES round function and variants taking advantage of the ARMv8 Cryptography Extension. Since AEGIS provides authenticated encryption directly, there is no need to apply a separate MAC.

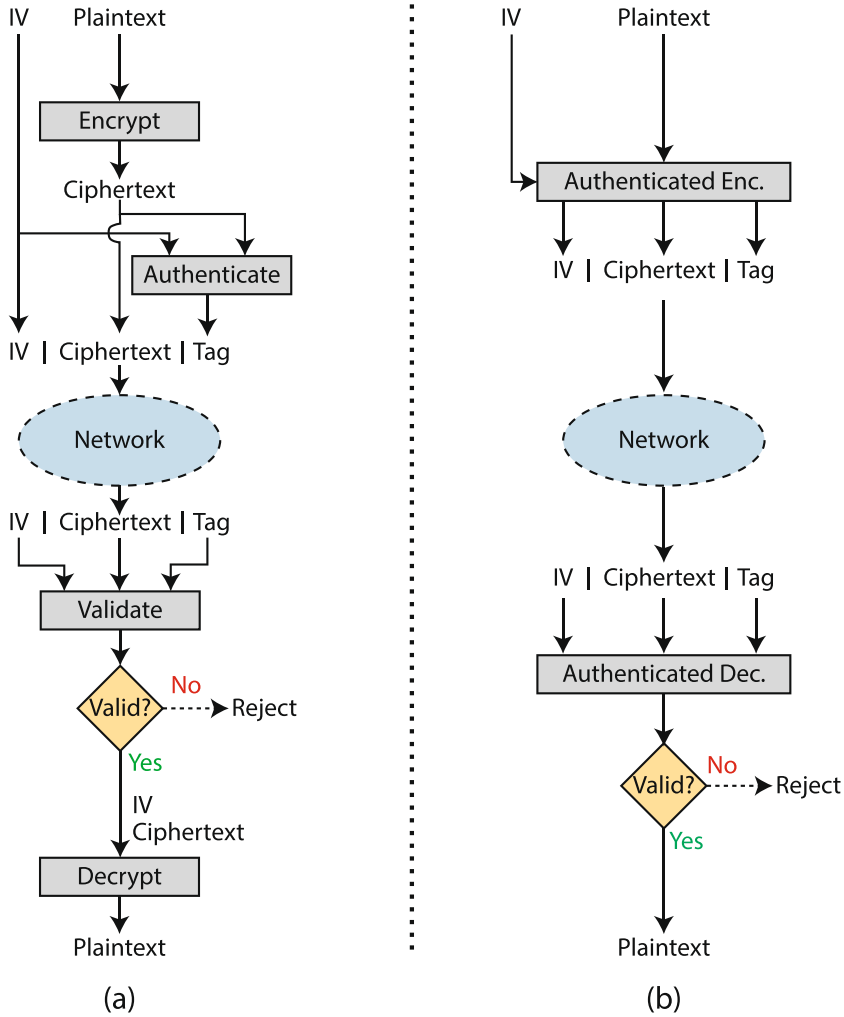


Fig. 3 **a** Authenticated encryption is obtained through an 'Encrypt-then-MAC'-composition. **b** Authenticated encryption is obtained through a dedicated authenticated encryption algorithm such as AEGIS

2.3 Compression Algorithms

Contrary to encryption, compression may or may not lead to loss of information. Accordingly, we classify compression algorithms as *lossy* or *lossless*. A compression algorithm is lossy if it induces a loss of information, while a lossless compression algorithm permits perfect reconstruction of the original data. When evaluating the performance of compression algorithms, metrics such as *data compression ratio*, *data quality*, and *computational complexity* are relevant. The data compression ratio measures the relative reduction in size produced by the compression algorithm. Lossy compression algorithms, e.g., Joint Photographic Experts Group (JPEG) [38], typically achieve high compression ratios ($>10{:}1$) without significantly reducing the data quality. Lossless compression algorithms such as Portable Network Graphics (PNG) [39] achieve much lower compression ratios, e.g., up to 4:1 [40]. The computational complexity of lossy compression algorithms is typically a function of the data size. In contrast, the computational complexity of lossless compression algorithms also depends on the entropy of the data. Therefore, we expect lossy compression algorithms to operate in constant time for the given data size, while we expect the time complexity of lossless compression algorithms to vary.

2.4 Implementations in ROS

To implement authenticated encryption in the ROS environment, the data field of the ROS message must be altered to make space for the ciphertext, IV, and the message tag, respectively. This can be done by resizing the data field. Once resized, the IV is placed at the front of the data field belonging to the image or point cloud message, respectively. The plaintext is encrypted and authenticated using either an authenticated encryption cipher such as AEGIS or the 'Encrypt-then-MAC' composition using a cipher and HMAC. As seen in Fig. 4, we only apply cryptographic operations on the data field belonging to the ROS message, thus leaving the header field unchanged before the complete ROS message is published to the ROS topic. Also, note

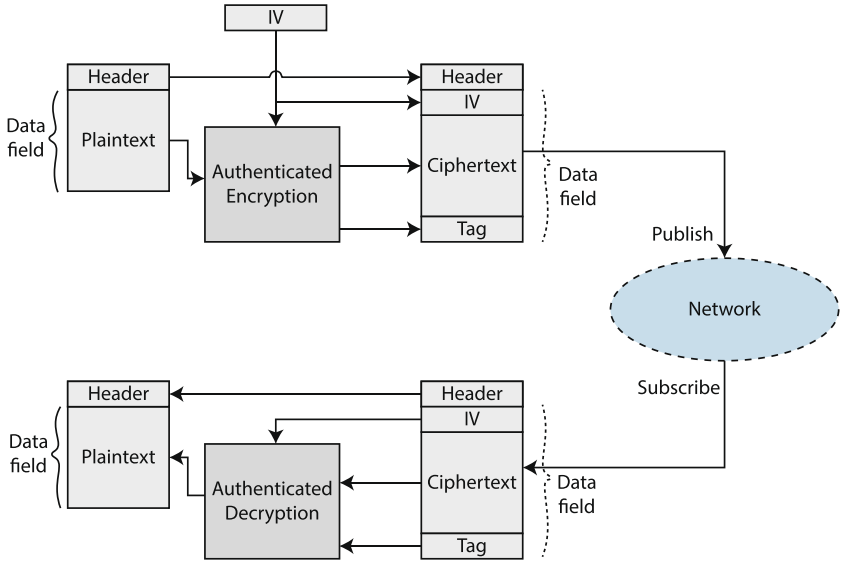


Fig. 4 The figure shows how the original ROS message is secured through the use of authenticated encryption. The data field belonging to the encrypted ROS message also makes space for the IV and the message tag

that AEGIS and the 'Encrypt-then-MAC' composition rely on different approaches to obtain authenticated encryption. Hence, they will differ in the way they are implemented in ROS. The pseudocode for the authenticated encryption and the authenticated decryption nodes when an 'Encrypt-then-MAC'-scheme is used can be seen in Algorithms 1 and 2, respectively.

Algorithm 1 Authenticated encryption node.

K: Symmetric key; IV: Initialization Vector;
$E_{K,IV}$: Encryption function parametrized by K and IV;
MAC_K: Message Authentication Code parametrized by K

Initialize MAC_K
while true **do**
  Initialize $E_{K,IV}$
  Plaintext ← Subscribe(Sensor Message)
  Ciphertext ← $E_{K,IV}$(Plaintext)
  Tag ← MAC_K(IV, Ciphertext)
  Secure Sensor Message ← (IV || Ciphertext || Tag)
  Publish(Secure Sensor Message)
  Update IV
end while

We implement image compression using lossless PNG and lossy JPEG compression, respectively. The compression algorithms are accessed through encoding/decoding functions in the OpenCV library. We use the ROS package *cv_bridge* to bridge ROS and OpenCV image data and then embed the encoding/decoding functions into the cryptographic pipeline. When combining compression and authenticated encryption, the order of the services plays an important role. For example, an 'encrypt-then-compress' scheme will result in low compression ratios because the ciphertext is very similar to white noise. In this scheme, lossless compression can be used but will result in very low compression ratios since there is no redundancy in white noise. Also, note that we cannot use lossy compression in such a scheme since it induces a loss of information. Consequently, the decryption algorithm is not given access to the original ciphertext and is therefore incapable of providing meaningful decryption. Therefore, we must apply the compression algorithms *before* the cryptographic operations. For this reason, we adopt a 'compress-then-encrypt' scheme in which authenticated encryption is applied to the output of the compression algorithm. We refer to the Github repository for any additional implementation-specific details [29].

Algorithm 2 Authenticated decryption node.

K: Symmetric key; IV: Initialization Vector;
$D_{K,IV}$: Decryption function parametrized by K and IV;
MAC_K: Message Authentication Code parametrized by K

Initialize MAC_K
while true **do**
  (IV' || Ciphertext' || Tag') ← Subscribe(Secure Sensor Message')
  Tag ← MAC_K(IV', Ciphertext')
  if Tag != Tag' **then**
    Reject Ciphertext'
  else
    Initialize $D_{K,IV'}$
    Plaintext' ← $D_{K,IV'}$(Ciphertext')
    Sensor Message' ← Plaintext'
    Publish(Sensor Message')
  end if
end while

3 Experimental Setup and Testing of Cryptographic Algorithms

Following the description of the cryptographic algorithms, compression algorithms, and implementation-specific details, we move over to experiments. We begin by describing the hardware, software, and experimental data. Then, we describe how each experiment was conducted and the obtained results. Finally, we make some remarks regarding the obtained results.

3.1 Hardware, Software, and Experimental Data

We perform the experiments on an Nvidia Jetson Xavier Developer Kit. The Jetson Xavier is an efficient edge-computing unit with a small form factor, applicable for autonomous vehicles. It delivers 93.75 GB/s of highspeed I/O, which eases the burden of handling large amounts of data. Furthermore, the effect is adjustable between 10W and 30W, depending on the power mode. To utilize full performance, we set the power mode to "MAXN" to activate

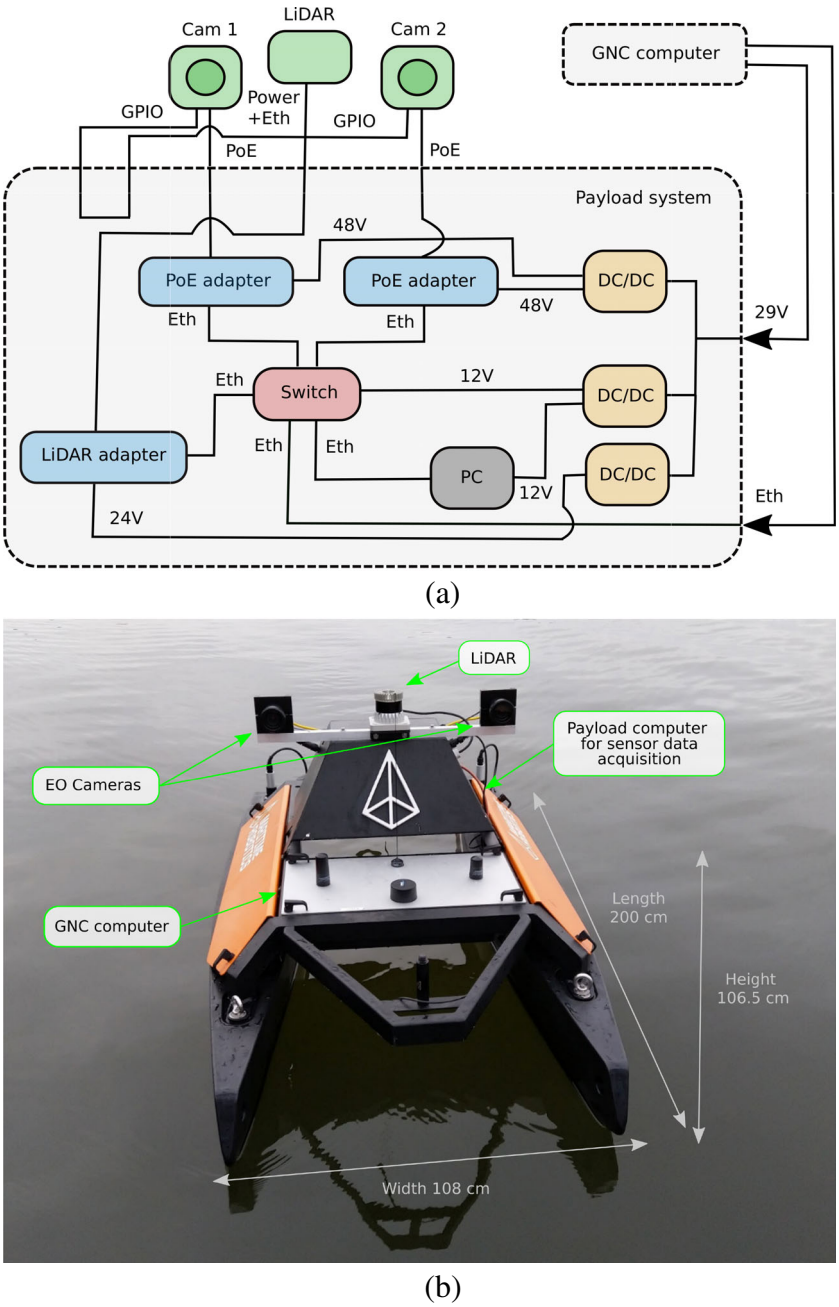
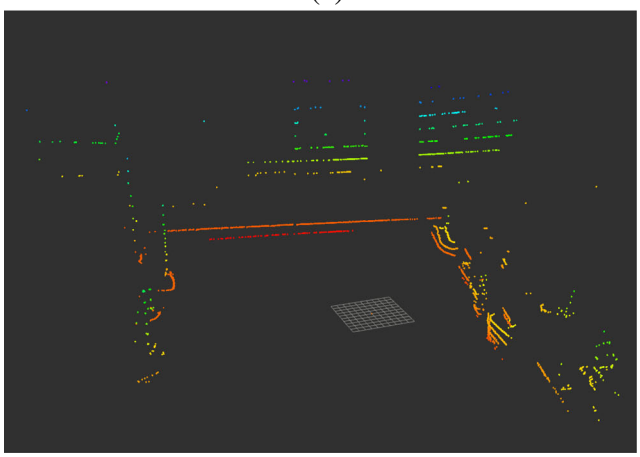


Fig. 5 **a** The figure gives an overview of the hardware architecture. **b** The Otter USV from Maritime Robotics is armed with two electro-optical (EO) cameras and a LiDAR for sensor data acquisition. The image is reproduced with kind permission of Maritime Robotics, www.maritimerobotics.com

all eight cores. We use Ubuntu 18.04 LTS and the g++ 7.5.0 compiler. The software tools used are ROS Melodic and OpenCV 3.4.3.

Images and point clouds were acquired using two electro-optical (EO) cameras and a LiDAR, respectively, onboard a small USV to test the algorithm performance on real-world data (see Fig. 5). The dataset contains image and point cloud data where the USV slowly navigates around the Trondheim Harbor (see Fig. 6). We use the Blackfly S GiGE camera with a 1280×1024 resolution. The images are recorded in monochrome pixel format at a frequency of 7.5 Hz. The LiDAR used is an Ouster OS-1 with a resolution of 2048×16 beams, running at 10 Hz. The data was recorded using the ROSbag tool to time stamp the data. The sensor data produced by the EO cameras and the LiDAR require a bandwidth of 9.75 MB/s and 31.5 MB/s, respectively. This data is then fed to the cryptographic pipeline to verify its value for real-world applications.

(a)



(b)

Fig. 6 The sensor data was recorded on-board an unmanned surface vehicle (USV) in the Trondheim Harbor. **a** shows a snapshot of the collected image data, and **b** shows a snapshot of the collected LiDAR data

3.2 Experiments and Results

The experimental setup consists of two Nvidia Jetson Xavier computers and one ethernet cable to enable wired, high-speed data transmission between the computing devices. To communicate and send data between nodes, ROS uses a master-slave setup. Only one node is assigned to be the master, and all other nodes must be configured to use this master. Using our analogy, the first computer, i.e., the master node, acts as the talker, while the second computer, i.e., the slave node, acts as the listener.

3.2.1 Experiment 1: Encryption Latency Measurements

In the first experiment, we are concerned with measuring the additional latency caused by encryption and decryption operations during the proposed cryptographic pipeline. We merge the latency caused by the encryption and decryption operations into one single cryptographic latency measurement, while network latency is ignored. Most of the computation is related to the core functionality accessed through initialization and processing. Minor parts relate to the allocation of input and output buffers and IV loading and incrementation for synchronization purposes. We refer to the public Github repository for additional details [29]. We benchmark AES in the CFB mode against the stream ciphers from the eSTREAM portfolio, i.e., HC-128, Sosemanuk, ChaCha20, and Rabbit. Both a table-driven (software-oriented) and a hardware-accelerated (ARMv8 Cryptography Extension) variant of AES CFB is benchmarked.

3.2.2 Results

Table 2 summarizes the cryptographic latency measurements for the encryption-only schemes on image and point cloud data across 1000 consecutive samples. These samples were used to calculate the mean and standard deviation. A visual representation of the results is shown in Fig. 7. Observe that the relative order between the algorithms is the same when comparing image and point cloud data. As shown, Rabbit produces the lowest latency closely followed by HC-128, while AES CFB produces the highest latency. In between, we find ChaCha20, Sosemanuk, and the hardware-accelerated variant of AES CFB, respectively.

3.2.3 Experiment 2: Authenticated Encryption Latency Measurements

The encryption algorithms assessed in Experiment 1 provide confidentiality only and do not ensure the integrity and authenticity of the message upon reception. Therefore, for

Table 2 The table shows a latency comparison for various encryption algorithms over 1000 samples related to experiment 1

Encryption algorithm	Mean latency [ms]	Std.Deviation latency [ms]
Image data size: 1.31 MB	μ	σ
Sosemanuk	8.0400	0.5935
ChaCha20	7.2184	0.3188
Rabbit	3.3055	0.5740
HC-128	3.9312	0.4858
AES CFB	14.6879	0.6558
AES CFB (HW accelerated)	10.6753	0.9205
Point cloud data size: 3.15 MB	μ	σ
Sosemanuk	18.6308	0.4184
ChaCha20	17.4546	0.5578
Rabbit	7.4948	0.4810
HC-128	9.3381	0.4608
AES CFB	33.3859	0.7673
AES CFB (HW accelerated)	26.3027	1.4332

Original works for the algorithms are found in Table 1

active attacks such as spoofing and message manipulation, we need to apply authenticated encryption. Consequently, we compare 'Encrypt-then-MAC' compositions with the authenticated encryption algorithm AEGIS in the second experiment. The 'Encrypt-then-MAC'-compositions consist of Rabbit and HC-128 composed with the HMAC-SHA-256 authentication algorithm, as they performed best in experiment 1. Both table-driven and hardware-accelerated variants of AEGIS are tested. As before, we merge the cryptographic operations into one single latency measurement.

3.2.4 Results

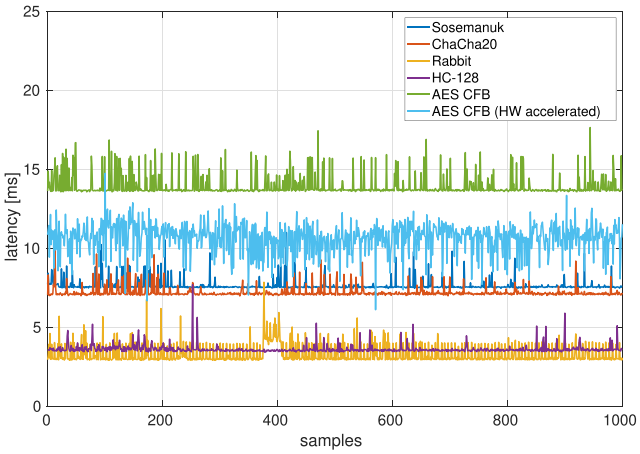
Table 3 summarizes the cryptographic latency measurements for the authenticated encryption schemes on image and point cloud data across 1000 consecutive samples. The samples were used to calculate the mean and standard deviation. A visual representation of the results is shown in Fig. 8. The hardware-accelerated variant of AEGIS proves to be the most efficient, followed by the table-driven AEGIS implementation, which is considerably faster than the HC-128+HMAC and Rabbit+HMAC schemes.

3.2.5 Experiment 3: Combining Image Compression and Cryptographic Operations

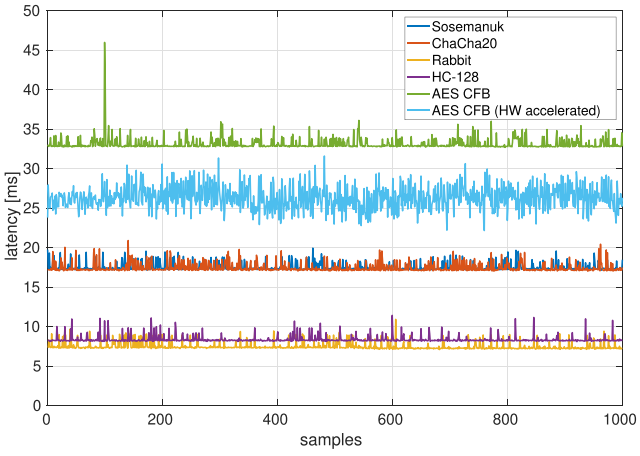
In the third experiment, we investigate whether compression before cryptographic operations is faster than cryptographic operations on the original data. Due to the remarkably efficient stream ciphers benchmarked so far, it is interesting to investigate if this is the case. Since we focus on authenticated encryption, and given the results from Experiment 2, AEGIS is used in a 'compress-then-encrypt' scheme. We assess the performance of 'compress-then-encrypt' schemes in which both lossy and lossless compression algorithms are composed with authenticated encryption. The lossy compression algorithm JPEG and the lossless compression algorithm PNG are used. The pipeline can be seen in Fig. 9.

3.2.6 Results

Table 4 summarizes the latencies based on image compression and cryptographic operations across 1000 consecutive samples. These samples were used to calculate the mean latency related to compression, cryptographic operations, and decompression, respectively. A visual representation of the results is shown in Fig. 10. On average, the original images were reduced by 74% and 30% when JPEG and PNG were used, respectively. Consequently, the cryptographic latency was reduced as well. However, this reduction is offset by the time it takes to perform compression and decompression. Note that lossless PNG compression and decompression produces significantly higher, and varying latencies than lossy JPEG, as expected.

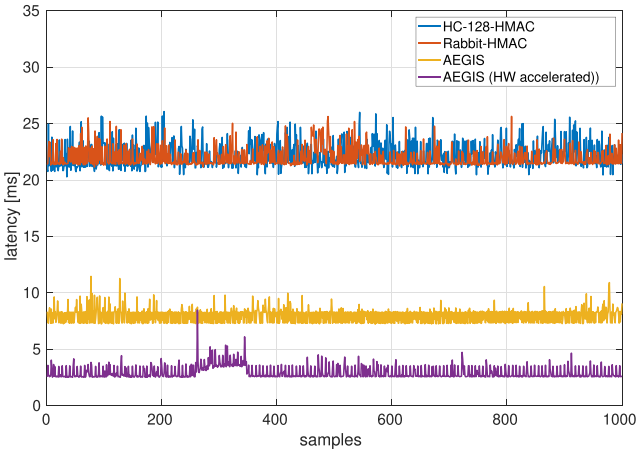


(a)

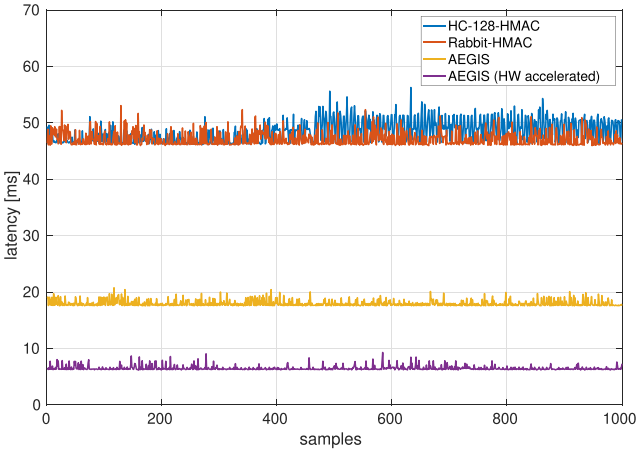


(b)

Fig. 7 **a** The figure shows the cryptographic latencies for image data across 1000 samples when encryption schemes are used in Experiment 1. **b** The figure shows the cryptographic latencies for point cloud data across 1000 samples when encryption schemes are used in Experiment 1



(a)



(b)

Fig. 8 **a** The figure shows the cryptographic latencies for image data across 1000 samples when encryption schemes are used in Experiment 2. **b** The figure shows the cryptographic latencies for point cloud data across 1000 samples when encryption schemes are used in Experiment 2

Table 3 The table shows a latency comparison for authenticated encryption algorithms over 1000 samples related to experiment 2

Authenticated encryption algorithm	Mean latency [ms]	Std.Deviation latency [ms]
Image data size: 1.31 MB	μ	σ
HC-128 + HMAC	22.3717	0.6925
Rabbit + HMAC	21.9876	0.8219
AEGIS	7.9323	0.6350
AEGIS (HW accelerated)	2.9235	0.5449
Point cloud data size: 3.15 MB	μ	σ
HC-128 + HMAC	48.3779	1.8759
Rabbit + HMAC	47.1431	1.2136
AEGIS	17.9901	0.5358
AEGIS (HW accelerated)	6.4980	0.3858

Original works for the algorithms are found in Table 1

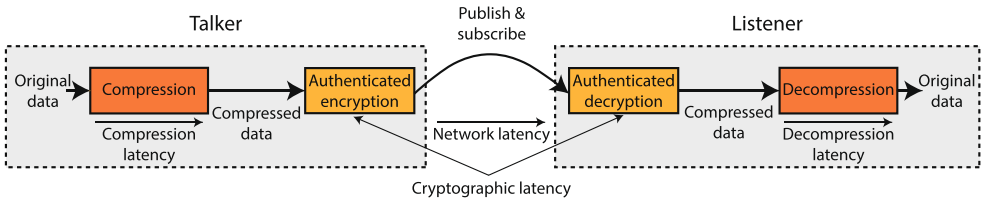


Fig. 9 The third experiment investigates a 'compress-then-encrypt' scheme where compression and authenticated encryption are combined. The latencies related to authenticated encryption and decryption operations, respectively, are merged into one measurement, i.e., the cryptographic latency

3.3 Remarks

The results show that the stream ciphers significantly outperform the AES block cipher, and we recommend the Rabbit and HC-128 stream ciphers if only confidentiality is required. However, when used in generic compositions, they are outperformed by the authenticated encryption algorithm AEGIS. We believe this is because the AEGIS algorithm derives a message tag from the internal state and does not require the instantiation and initialization of a separate MAC, which is the case for the generic compositions. As such, we recommend that AEGIS is used when authenticated encryption is required.

The use of compression and decompression reduces the data size and, subsequently, the cryptographic latency. However, this gain is offset by the latency induced by compression and decompression, as seen in Experiment 3. The latency induced by the PNG compression was considerable, even at a relatively low compression ratio. While smaller compression ratios would result in reduced compression time, the size of the compressed image rapidly converges to the size of the original image. As such, we find PNG compression to be unsuitable for time-critical applications. Regarding JPEG, we find that the total latency of JPEG + AEGIS is higher than if AEGIS is applied directly. However, JPEG also reduces the bandwidth required significantly and might therefore be considered if bandwidth is constrained. Interestingly, if less efficient cryptographic schemes are used, e.g., AES CFB + HMAC, lossy compression may be beneficial. That is, the 'compress-then-encrypt' scheme may produce lower total latency than if cryptographic algorithms are applied directly. This is significant since it implies that the cryptographic algorithms proposed by previous work, e.g., [24–26], could benefit from lossy compression algorithms. This is no longer the case when AEGIS is used.

Another aspect to consider is the data quality. Since the data used in guidance, navigation, and control must be of high quality, i.e., near-lossless, we set the compression ratio low. Due to the low compression ratio, the Huffman encoding step of the JPEG algorithm must process a relatively large amount of data with $O(n \log n)$ run time [42]. This step is believed to be the bottleneck of the pipeline. With higher compression ratios, the JPEG algorithm is faster but never faster than using AEGIS directly. Additionally, increasing compression ratios will progressively deteriorate the data and thereby the performance of the GNC system.

4 Conclusions

With the increased use of vision-based sensors such as cameras and LiDARs in autonomous vehicles, it is essential to consider how the signals from these sensors can be secured efficiently. The vision-based signals pose a significant challenge because they require much greater throughput than traditional signals in feedback control. Previous research on how vision-based feedback control signals can be secured has been restricted to the use of block ciphers, which are much less efficient.

We address this problem by suggesting modern stream ciphers and demonstrate that these ciphers perform much better than the block ciphers proposed by previous work. We have also demonstrated that AEGIS gives the best results if authenticated encryption is required. Finally, we find that while compression may accelerate the cryptographic pipeline for algorithms proposed by other authors, this is no longer the case when AEGIS is used. As a result, compression algorithms should only be combined with AEGIS if network bandwidth is constrained. All algorithms have been implemented in ROS and made publicly available through a Github repository [29]. In the future, we plan to implement and conduct full-scale experiments to show that the proposed method indeed applies to more resource-demanding feedback control applications.

Table 4 Latency comparison for PNG+AEGIS and JPEG+AEGIS in which both are benchmarked up against AEGIS on raw image data, i.e., no compression, using 1000 samples

Authenticated encryption algorithm	Compression algorithm	Mean compressed data size [MB]	Mean latency compression [ms]	Mean cryptographic latency [ms]	Mean decompression [ms]	Mean total latency [ms]
Image data size: 1.31 MB		n	μ_1	μ_2	μ_3	μ
AEGIS	JPEG[a]	0.31	11.5161	1.9274	6.7596	20.2037
AEGIS	PNG[b]	0.92	137.0573	5.3907	21.0929	163.5409
AEGIS	–	–	–	7.9323	–	7.9323

The original work of AEGIS is found in Table 1

[a]JPEG compression level is set to default value according to the OpenCV specification [41], i.e., 95/100

[b]PNG compression level is set to default value according to the OpenCV specification [41], i.e., 3/9

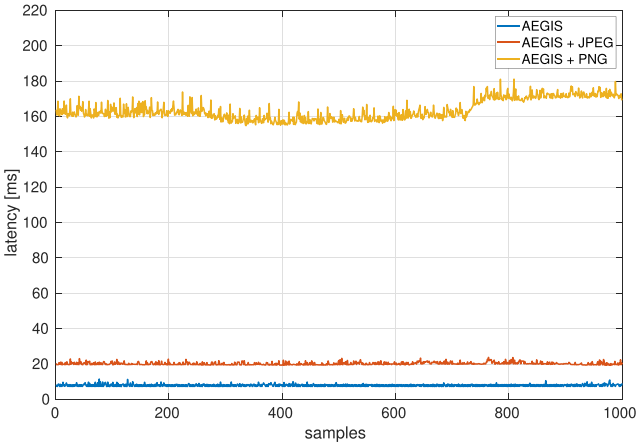


Fig. 10 Overall latencies when image data is compressed, encrypted, decrypted, and decompressed across 1000 samples in Experiment 3. Here, lossless PNG and lossy JPEG compression are combined with AEGIS to reduce the amount of image data to be encrypted, transmitted, and decrypted

Acknowledgements This work was supported by the Norwegian Research Council (project no. 223254) through the NTNU Center of Autonomous Marine Operations and Systems (AMOS) at the Norwegian University of Science and Technology.

Author Contributions The first author, Øystein Volden, has contributed to software development, integration of cryptographic algorithms into ROS, data collection, and the experimental setup. He also contributed to the first, second, and third drafts of the manuscript, including the preparation of relevant material and analysis. The second author, Petter Solnør, has contributed to software development and integration of the cryptographic algorithms into ROS. Also, he has contributed to the first, second, and third drafts of the manuscript, including the preparation of relevant material and analysis. The third author, Slobodan Petrovic, has contributed with valuable discussions regarding the cryptographic aspects and proof-reading. The fourth author, Thor I. Fossen, has contributed with valuable discussions of concepts regarding security in guidance, navigation, and control, as well as proof-reading. All authors read and approved the revised manuscript.

Funding Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital). This work was funded by the Norwegian Research Council (project no. 223254) through the NTNU Center of Autonomous Marine Operations and Systems (AMOS) at the Norwegian University of Science and Technology.

Availability of data and materials The data that support the findings of this study are openly available in the public Github repository "Crypto ROS: Secure and Efficient Transmission of Vision-Based Feedback Control Signals" [29].

Declarations

Ethics approval No ethical approval was deemed necessary.

Consent to participate The authors, Øystein Volden, Petter Solnør, Slobodan Petrovic, and Thor I. Fossen, voluntarily agree to participate in this research study.

Consent for Publication The authors, Øystein Volden, Petter Solnør, Slobodan Petrovic, and Thor I. Fossen, give their consent for information about themselves to be published in the Journal of Intelligent & Robotic Systems. We understand that the text and any pictures or videos published in the article will be used only in educational publications intended for professionals, or if the publication or product is published on an open access basis. We understand that it will be freely available on the internet and may be seen by the general public. We understand that the pictures and text may also appear on other websites or in print, may be translated into other languages or used for commercial purposes. We understand that the information will be published without our child's name attached, but that full anonymity cannot be guaranteed. We have been offered the opportunity to read the manuscript. We acknowledge that it is not possible to ensure complete anonymity, and someone may be able to recognize me. However, by signing this consent form we do not in any way give up, waive or remove my rights to privacy. I may revoke my consent at any time before publication, but once the information has been committed to publication ("gone to press"), revocation of the consent is no longer possible.

Competing interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Felski, A., Zwolak, K.: The ocean-going autonomous ship—challenges and threats. Journal of Marine Science and Engineering 8(1). https://doi.org/10.3390/jmse8010041. https://www.mdpi.com/2077-1312/8/1/41 (2020)
2. Bolbot, V., Theotokatos, G., Boulougouris, E., Vassalos, D.: A novel cyber-risk assessment method for ship systems. Saf. Sci. **131**, 104908 (2020). https://doi.org/10.1016/j.ssci.2020.104908. https://www.sciencedirect.com/science/article/pii/S0925753520303052
3. Fossen, T.I. Handbook of Marine Craft Hydrodynamics and Motion Control, 2nd. Wiley, New York (2021)
4. Zhang, X., Han, Q., Ge, X., Ding, D., Ding, L., Yue, D., Peng, C.: Networked control systems: a survey of trends and techniques. IEEE/CAA J. Autom. Sinica **7**(1), 1–17 (2020). https://doi.org/10.1109/JAS.2019.1911651
5. Hespanha, J.P., Naghshtabrizi, P., Xu, Y.: A survey of recent results in networked control systems. Proc. IEEE **95**(1), 138–162 (2007)
6. Zhao, X., Sun, P., Xu, Z., Min, H., Yu, H.: Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications. IEEE Sensors J. **20**(9), 4901–4913 (2020). https://doi.org/10.1109/JSEN.2020.2966034
7. Song, H., Lee, K., Kim, D.H.: Obstacle avoidance system with lidar sensor based fuzzy control for an autonomous unmanned ship. In: 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), pp. 718–722. https://doi.org/10.1109/SCIS-ISIS.2018.00119 (2018)
8. Trslic, P., Rossi, M., Robinson, L., O'Donnel, C.W., Weir, A., Coleman, J., Riordan, J., Omerdic, E., Dooly, G., Toal, D.: Vision based autonomous docking for work class rovs. Ocean Eng. **196**, 106840 (2020). https://doi.org/10.1016/j.oceaneng.2019.106840. https://www.sciencedirect.com/science/article/pii/S0029801819309369
9. Ji, K., Kim, W.J.: Real-time control of networked control systems via ethernet. International Journal of Control Automation and Systems 3 (2004)
10. Teixeira, A., Pérez, D., Sandberg, H., Johansson, K.H.: Attack models and scenarios for networked control systems. In: Proceedings of the 1st international conference on high confidence networked systems, HiCoNS '12, pp 55–64, Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2185505.2185515 (2012)
11. Wang, Q., Yang, H.: A survey on the recent development of securing the networked control systems. Syst. Sci. Control Eng. **7**(1), 54–64 (2019). https://doi.org/10.1080/21642583.2019.1566800
12. National Bureau of Standards: Data Encryption Standard (DES). Federal Information Processing Standards Publication 46 (1977)
13. Barker, E., Mouha, N.: Recommendation for the triple data encryption algorithm (TDEA) block cipher. https://doi.org/10.6028/NIST.SP.800-67r2 (2017)
14. National Institute of Standards and Technology: Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197 (2001)
15. Rivest, R.: The MD5 message-digest algorithm. RFC 1321 RFC Editor (1992)
16. Dang, Q.H.: The keyed-hash message authentication code (HMAC) - FIPS 198-1. Tech. rep., Gaithersburg, MD USA (2008)
17. Gupta, R.A., Chow, M.: Performance assessment and compensation for secure networked control systems. In: 2008 34th Annual Conference of IEEE Industrial Electronics, pp. 2929–2934 (2008)
18. Pang, Z., Liu, G.: Design and implementation of secure networked predictive control systems under deception attacks. IEEE Trans. Control Syst. Technol. **20**(5), 1334–1342 (2012)
19. Jithish, J., Sankaran, S.: Securing networked control systems: Modeling attacks and defenses. In: 2017 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), pp. 7–11 (2017)
20. An, J.H., Bellare, M.: Does encryption with redundancy provide authenticity?. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001, pp. 512–528. Springer, Berlin (2001)
21. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. J. Cryptol. **21**(4), 469–491 (2008). https://doi.org/10.1007/s00145-008-9026-x
22. Teixeira, R.R., Maurell, I.P., Drews, P.L.J.: Security on ROS: analyzing and exploiting vulnerabilities of ROS-based systems. In: 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), pp 1–6. https://doi.org/10.1109/LARS/SBR/WRE51543.2020.9307107 (2020)
23. Schneier, B.: Description of a new variable-length key, 64-bit block cipher (blowfish). In: International Workshop on Fast Software Encryption, pp. 191–204. Springer (1993)

24. Lera, F.J.R., Balsa, J., Casado, F., Fernández, C., Rico, F.M., Matellán, V.: Cybersecurity in autonomous systems: Evaluating the performance of hardening ROS. Málaga, Spain 47 (2016)
25. Rodríguez-Lera, F.J., Matellán-Olivera, V., Balsa-Comerón, J., Guerrero-Higueras, Á. M., Fernández-Llamas, C.: Message encryption in robot operating system: Collateral effects of hardening mobile robots. Frontiers in ICT **5**, 11 (2018). https://doi.org/10.3389/fict.2018.00002
26. Balsa-Comerón, J., Guerrero-Higueras, Á.M., Rodríguez-Lera, F.J., Fernández-Llamas, C., Matellán-Olivera, V.: Cybersecurity in autonomous systems: Hardening ROS using encrypted communications and semantic rules. In: Ollero, A., Sanfeliu, A., Montano, L., Lau, N., Cardeira, C. (eds.) ROBOT 2017: Third Iberian Robotics Conference, pp. 67–78. Springer International Publishing, Cham (2018)
27. Robshaw, M.: The eSTREAM Project, pp. 1–6. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-68351-3_1
28. Wu, H., Preneel, B.: AEGIS: a fast authenticated encryption algorithm. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) Selected Areas in Cryptography – SAC 2013, pp. 185–201. Springer, Berlin (2014)
29. Volden, Ø., Solnør, P.: Crypto ROS: Secure and Efficient Transmission of Vision-Based Feedback Control Signals. https://github.com/oysteinvolden/Real-time-sensor-encryption (2020)
30. Wu, H.: The Stream Cipher HC-128. In: The eSTREAM Finalists (2008)
31. Bernstein, D.: ChaCha, a variant of Salsa20. https://cr.yp.to/chacha/chacha-20080120.pdf (2008)
32. Boesgaard, M., Vesterager, M., Zenner, E.: The Rabbit Stream Cipher, pp. 69–83. Springer-Verlag, Berlin (2008)
33. Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Gouget, A., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: Sosemanuk, a fast software-oriented stream cipher, pp. 98–118. Springer-Verlag, Berlin (2008)
34. Quynh, H.: Dang: Secure hash standard - federal information processing standard publication 180-4. Tech. rep., Gaithersburg, MD USA (2015)
35. Solnør, P.: A cryptographic toolbox for feedback control systems. Model. Identif, Control **41**(4), 313–332 (2020). https://doi.org/10.4173/mic.2020.4.3
36. Turner, S., Chen, L.: Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 algorithms. RFC 6151 (2011)
37. Rogaway, P., Bellare, M., Black, J.: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. (TISSEC) **6**(3), 365–403 (2003)
38. Wallace, G.K.: The JPEG still picture compression standard. IEEE Trans. Consum. Electron. **38**(1), xviii–xxxiv (1992). https://doi.org/10.1109/30.125072
39. Boutell, T.: e.a.: RFC 2083: PNG (Portable Network Graphics) Specification (1997)
40. ZainEldin, H., Elhosseini, M.A., Ali, H.A.: Image compression algorithms in wireless multimedia sensor networks: A survey. Ain Shams Eng. J. **6**(2), 481–490 (2015). https://doi.org/10.1016/j.asej.2014.11.001. https://www.sciencedirect.com/science/article/pii/S2090447914001567
41. Image file reading and writing (2018). https://docs.opencv.org/3.4.3/d4/da8/group__imgcodecs.html. Accessed: 2020-06-20
42. Huffman, D.A.: A method for the construction of minimum-redundancy codes. Proc. IRE **40**(9), 1098–1101 (1952). https://doi.org/10.1109/JRPROC.1952.273898

Øystein Volden received the MSc degree in Engineering Cybernetics from the Norwegian University of Science and Technology (NTNU) in 2020. He is currently a PhD candidate in Engineering Cybernetics at NTNU and is affiliated with the NTNU Centre for Autonomous Marine Operations and Systems. He works with topics related to computer vision, machine learning, and cybersecurity for unmanned surface vehicles.

Petter Solnør obtained his MSc degree in Engineering Cybernetics from the Norwegian University of Science and Technology (NTNU) in 2021. He is currently a PhD candidate at the Department of Engineering Cybernetics at NTNU and is affiliated with the NTNU Centre for Autonomous Marine Operations and Systems. He works on topics related to applied cryptography and cybersecurity in unmanned surface vehicles.

Slobodan Petrovic obtained his PhD degree from University of Belgrade, Serbia in 1994. He worked at Institute of Applied Mathematics and Electronics and Institute of Mathematics in Belgrade from 1986 to 2000. He also worked on various information security-related projects at Institute of Applied Physics, Madrid, Spain, from 2000 to 2004. From 2004 to 2015, he was with Gjøvik University College, Norway, and since January 1st, 2016, he is professor of information security at Norwegian University of Science and Technology (NTNU), where he teaches cryptology and intrusion detection and prevention. His research interests include cryptology, intrusion detection, and digital forensics. He is author of more than 50 scientific papers from the field of information security, digital forensics, and cryptology.

Thor I. Fossen is a naval architect and a cyberneticist. He received an MSc degree in Marine Technology in 1987 and a PhD degree in Engineering Cybernetics in 1991, both from the Norwegian University of Science and Technology (NTNU). He is currently a professor of guidance, navigation, and control. Fossen's expertise covers guidance systems, inertial navigation systems, nonlinear control and observer theory, vehicle dynamics, hydrodynamics, autopilots, and unmanned vehicles. He has authored three Wiley textbooks. Fossen is one of the co-founders and former Vice President R&D of the company Marine Cybernetics AS, which DNV acquired in 2012. He is also co-founder of SCOUT Drone Inspection AS (2017). He received the Automatica Prize Paper Award in 2002 and the Arch T. Colwell Merit Award in 2008 at the SAE World Congress. He has been elected to the Academy of Technological Sciences (1998) and elevated to IEEE Fellow (2016).