frontiers in ICT

ORIGINAL RESEARCH
published: 02 March 2018
doi: 10.3389/fict.2018.00002

Message Encryption in Robot Operating System: Collateral Effects of Hardening Mobile Robots

Francisco J. Rodríguez-Lera[1], Vicente Matellán-Olivera[2], Jesús Balsa-Comerón[2], Ángel Manuel Guerrero-Higueras[2] and Camino Fernández-Llamas[2]*

[1] *AI Robolab, Computer Science and Communications Research Unit, Faculty of Science, Technology and Communications, University of Luxembourg, Esch-sur-Alzette, Luxembourg,* [2] *Robotics Group, Research Institute on Applied Sciences in Cybersecurity, Universidad de León, León, Spain*

In human–robot interaction situations, robot sensors collect huge amounts of data from the environment in order to characterize the situation. Some of the gathered data ought to be treated as private, such as medical data (i.e., medication guidelines), personal, and safety information (i.e., images of children, home habits, alarm codes, etc.). However, most robotic software development frameworks are not designed for securely managing this information. This paper analyzes the scenario of hardening one of the most widely used robotic middlewares, Robot Operating System (ROS). The study investigates a robot's performance when ciphering the messages interchanged between ROS nodes under the publish/subscribe paradigm. In particular, this research focuses on the nodes that manage cameras and LIDAR sensors, which are two of the most extended sensing solutions in mobile robotics, and analyzes the collateral effects on the robot's achievement under different computing capabilities and encryption algorithms (3DES, AES, and Blowfish) to robot performance. The findings present empirical evidence that simple encryption algorithms are lightweight enough to provide cyber-security even in low-powered robots when carefully designed and implemented. Nevertheless, these techniques come with a number of serious drawbacks regarding robot autonomy and performance if they are applied randomly. To avoid these issues, we define a taxonomy that links the type of ROS message, computational units, and the encryption methods. As a result, we present a model to select the optimal options for hardening a mobile robot using ROS.

Keywords: cyber-security, robot, ROS, sensors, encryption, empirical analysis, robot hardening, robots in public spaces

OPEN ACCESS

Edited by:
Frank Kargl,
University of Ulm, Germany

Reviewed by:
Henning Kopp,
University of Ulm, Germany
Stefan Dietzel,
Humboldt-Universität zu Berlin, Germany
Oswald Berthold,
Humboldt-Universität zu Berlin, Germany

****Correspondence:***
Francisco J. Rodríguez-Lera
francisco.lera@uni.lu

Specialty section:
This article was submitted to Computer and Network Security, a section of the journal Frontiers in ICT

Received: *02 October 2017*
Accepted: *05 February 2018*
Published: *02 March 2018*

Citation:
Rodríguez-Lera FJ, Matellán-Olivera V, Balsa-Comerón J, Guerrero-Higueras ÁM and Fernández-Llamas C (2018) Message Encryption in Robot Operating System: Collateral Effects of Hardening Mobile Robots. Front. ICT 5:2. doi: 10.3389/fict.2018.00002

1. INTRODUCTION

Deploying robots in real public spaces, such as schools, hospitals, or day-care centers, requires human–robot interaction capabilities, which means that robots listen, talk, record, and interact with people. It also means that robots process personal data from their cameras, microphones, or laser sensors. This information can be used, for instance, to identify the person interacting with the robot, which raises several privacy concerns and leads to robots not being compliant with current laws, such as the EU General Data Protection Regulation (GDPR).[1]

[1] https://www.eugdpr.org/

Furthermore, it is necessary to understand the security issues involved when moving a robot from an isolated lab to a public space. It is complicated for some robotics researchers be aware of the security problems associated with their work when they are focused on well-defined topics (for instance, when they are improving a manipulation algorithm). Thus, the deployment of their contributions is out of their scope, so the security issues related to the infrastructure used by robots (Morante et al., 2015).

Several vectors of attack on robotic systems have been described in Denning et al. (2009) and can be extremely important. To mention only one issue: the implications of a surgery robot suffering a cyber-attack could put a patient's life in danger. Two concerns, privacy and security, ought to be at the core of software development for mobile robots. It is possible to solve security issues by using COTS (Commercial Off-The-Shelf) libraries, or open sources, but they are not usually designed for robotic platforms due to added network overhead or CPU consumption that could affect the performance of the system. This, in turn, can affect the robot decision-making algorithms, resulting in erratic robotic behaviors that are not acceptable. This fact is especially relevant in robots interacting with humans, which can result in bodily harm and/or privacy invasion.

This study empirically analyzes the effects of solving privacy and security issues in mobile robots when applying ciphering to part of the information managed by the robot. In particular, it analyzes how the performance of different software components is affected. An additional contribution is the characterization of the different solutions that should be used taking into account the hardware and data manipulation capabilities of the different robots.

This method for interchanging messages presents the worst scenario for a robot deployed in a public space. It may suffer eavesdropping, spoofing, or denial-of-service attacks. Besides, unauthorized intruders may compromise the privacy of the users by accessing the cameras of the robot, any of its sensor readings, or by tracking a person.

This paper proposes solving these problems by ciphering TCPROS messages. In particular, it analyzes the behavior of three well-known encryption algorithms (3DES, AES, and Blowfish) and their effects on the autonomy of the robot. The results show that a trade-off between the encryption algorithm and the robot autonomy is needed, encompassing the type of sensors used in the robot, level of security required, the volume of data exchanged among ROS nodes, and the computer capabilities of the robot.

The remainder of this paper is organized as follows: section 2 presents the fundamentals of the solution proposed and the related work. Section 3 describes the methodology of the experiments carried out to evaluate our proposal. Section 4 shows the empirical results and section 5 discusses the trade-offs between performance and security. Section 6 summarizes the main contributions made and future work envisioned.

2. BACKGROUND

Autonomous cars, vacuum cleaners at homes, robot-care assistance have a computer-based nature. This characteristic presents robots as computers that can suffer the same cyber-attacks and, therefore, to apply some degree of cyber-security (Morante et al., 2015). Basically, both platforms have operative systems and applications, however, the robot copes major control and management features introducing the concept of distributed frameworks.

Robot Operative System also known as ROS has become the most popular distributed framework for developing robotic applications (Quigley et al., 2009). It started as a research project, but currently, most manufacturers of commercial platforms use ROS for building robotic applications. Thus, ROS has become the *de facto* standard for robotic software development. ROS includes a set of libraries for controlling robots similar to operating system services, providing hardware abstraction for sensors and actuators, low-level device control, and inter-process communication. Computation takes place in ROS processes named Nodes. ROS implements a message-passing distributed system based on the publish/subscription paradigm, in which Nodes publish Messages on Topics which other Nodes consume. Messages are distributed using TCPROS, a transport layer for ROS Messages and Services. It uses standard TCP/IP sockets for transporting message data. Inbound connections are received via a TCP Server Socket with a header containing information about the message such as data type, routing information, the name of the Node sending data, the name of the Topic where the subscriber is connecting to, etc. Unfortunately, the transmission of messages was implemented in plain-text, so it is easy to access robot data or to modify robot behavior just by connecting to the same network. **Figure 1** presents the model illustrated in ROS Technical Overview (ROS Wiki, 2014).

By July 2017[2] ROS binary packages had been downloaded 13,441,711 times, counting only downloads from the main repository, not from the numerous mirrors around the globe. However, this scenario of massive public acceptance and developer-friendly software ecosystem has not led to a guide for secure development. On the contrary, it is possible to identify major security problems in the current version of ROS: plain-text communications, unprotected TCP ports, XML-RPC legacy issues, and unencrypted data storage.

ROS security issues can be exploited by malicious software in several ways, as shown in McClean et al. (2013), where a ROS-based honeypot was set to find vulnerabilities in a mobile robot. However, few solutions have been proposed. In Huang et al. (2014), a runtime verification framework for robotic applications built on ROS is presented to address the safety and security issues of robots. A transparent monitoring infrastructure is provided for intercepting and monitoring the commands and messages passing through the system. However, privacy problems caused by plain messages being eavesdropped are not tackled by this solution, and the authors acknowledged latency between messages due to the runtime monitoring process, but no measurements of these problems were reported in their work.

Vuong et al. (2015) experimentally evaluated a decision tree-based method for detecting cyber-attacks on a robotic vehicle in a variety of scenarios involving denial of service, command

[2](2016a). *ROS Community Metrics Report*. Available at: http://download.ros.org/downloads/metrics/metrics-report-2016-07.pdf.

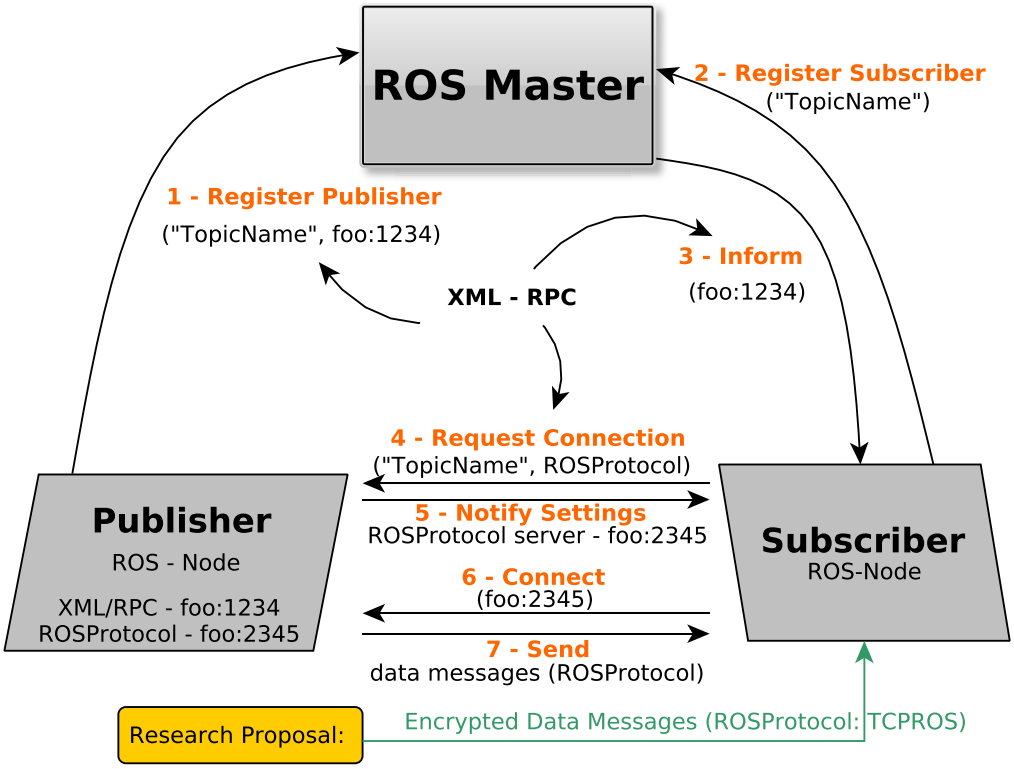

FIGURE 1 | Establishing connections through the ROS Topic paradigm. (Derivative of "Establishing a topic connection" by (ROS Wiki, 2014) which is licensed under CC BY 3.0).

injection, and two types of malware attacks. In the same way, Sabaliauskaite et al. (2016) describe an experimental evaluation of a method to identify the interrelations between robot failures and cyber-attacks. Two types of cyber-attacks on sensor measurements were implemented: false data injection and scaling. Experiments analyzed if the robot failed to complete the task of reaching the destination. We propose a similar approach, in which we study the influence of using encrypted data exchanged in the communication between two ROS nodes using different hardware configurations and distinct sensors.

Three main working approaches have been proposed to face these security issues. The first one is the development of a new version of ROS: ROS2 (Hood and Woodall, 2016). This approach aims to solve some of the security issues by using the Distributed Data System (DDS) library instead of TCPROS. Unfortunately, ROS2 is still a Beta solution. The second approach is based on the use of traditional Kernel security tools. It is centered around the development of SROS (White et al., 2016) and uses TLS (Breiling et al., 2017) to secure communications. However, SROS is a highly experimental solution and still under heavy development,[3] and it has not been used in any robot in production until now. The third line groups *ad hoc* solutions that face one or multiple security issues at the application level rather than at kernel or network level. This article may fall into the third category. It is widely recognized that COTS libraries are good options for solving security problems; however, these approaches have collateral implications which have to be studied.

Privacy problems in robotics communications have already been dealt with in specific domains. For example, a secure communications standard called the "Interoperable Tele-surgery Protocol" (ITP) has been proposed for surgical robotics. This standard specifies how the master and slave components communicate and its security has been taken into account, as seen in Lee and Thuraisingham (2012). However, in this domain, dedicated communication channels are taken for granted and no performance problems are expected. Besides, specific domains can make strong assumptions about the equipment involved, but in a general framework such as ROS, hardware capabilities, type of sensors, etc. cannot be anticipated.

The authors (Lera et al., 2016) have proposed hardening the TCPROS protocol by encrypting some fields of the data messages exchanged. However, the first approach was focused on only one algorithm and it did not offer much of a solution for the problems found. There are many alternatives in the literature for encrypting communications. Symmetric (private) key encryption methods seem to be the best-suited ones according to Hardjono and Dondeti (2005), particularly those accepted by the National Institute of Standards and Technology (NIST) (Barker and Barker, 2016).

We have selected two block symmetric cipher algorithms recommended by the National Institute of Standards and Technology

[3](2016b). *SROS a Set of Security Enhancements for ROS*. Available at: http://wiki.ros.org/SROS.

(NIST) (Barker and Barker, 2016): Triple Data Encryption Algorithm (also known as 3DES, 3TDES, or 3TDEA), and Advanced Encryption Standard (AES). In addition, we have also considered the Blowfish algorithm, because it is recognized as one of the most efficient algorithms considering time consumed for encryption and memory needed for implementation (Patil et al., 2016).

3DES is an encryption algorithm that uses a 64-bit block size and a 192-bit key size. It is similar to the one in the original DES (Smid and Branstad, 1988). The approach proposes applying the DES encryption process 3 times in order to improve the encryption level. AES (Daemen and Rijmen, 2013) is a symmetric key block cipher that has a variable key length of 128, 192, or 256 bits. It is able to encrypt data-blocks of 128 bits in 10, 12, and 14 rounds depending on the key size. Each round consists of a substitution and a permutation, and the whole construction is called a substitution–permutation network. By contrast to, e.g., 3DES which is a Feistel network. Due to AES flexibility (Gueron, 2010), it is suitable for both hardware and software implementation. Blowfish (Schneier, 1993) uses a 64-bit block and allows a variable-length key, ranging from 32 to 448 bits.

The performance of these algorithm methods has been extensively studied by Nadeem and Javed (2005) and Elminaam et al. (2008). It has been proven that Blowfish has better performance, for instance, for encrypting videos or sounds (Elminaam et al., 2010), but in other situations, AES shows better results (Schneier et al., 1999). Therefore, an empirical approach has been chosen to try to decide which algorithm is more suitable, depending on the configuration and capabilities of the robot.

3. MODEL, MATERIAL, AND METHODS

This section defines the characteristics of a cyber-attack to a mobile robotic platform deployed in a real environment. Besides, it describes the materials (hardware and software) used to implement the robot hardening and the methodology used to evaluate the robot performance under hardening circumstances.

3.1. Cyber-Attack Model

Distributed properties of ROS framework allow us to model our approach inspired by the Raymond (2001) attacker classification applied to a network. It covers three factors: location, local: internal–external, the attacker is attacking the robot, in the same network of the robot or behind the cloud; behavior, passive–active, the attacker actively changes the status of the robot or remain passive gathering data; and attitude, static-adaptive, the attacker compromises a robot resource and then changes its behavior during the attack execution or not.

Moreover, framing the attacks in a classic cyber-security taxonomy present a robot facing three basic types of issues: availability (data interruption), one or multiple sensors are not working properly; confidentiality (data capturing), the data from robot sensor is being watched by the attacker; and data integrity (data modification), data from a sensor is modified lying to the decision-making system. In addition, a robot can also suffer authenticity and non-repudiation issues, giving as a result data fabrication from a non-trusted origin (Sattarova Feruza and Kim, 2007). As a result of an attack, the robot can perform undesired behaviors or offer false information to robot users, interrupt partial or totally the robot behavior as well as to offer all the information available in the scenario to an attacker. Moreover, these attacks can be correlated and extended with the classical security issues proposed by authors such as Panchenko and Pimenidis (2006): denial of service, replay attack, analyze application layer data,… **Figure 2** summarizes graphically these issues in a real robotic environment.

This study proposes to protect the robot under those attacks whose aim is to generate undesired robot behaviors or to gather scenario information from a robot using ROS in a real human–robot scenario. The main consequences of this kind of attack are two: the safety concerns generated in the human–robot scenario and the privacy concerns given human interaction.

To encrypt data transmitted between ROS processes is a valid response to face an attacker connected to the same network (internal attack) with a passive–active behavior and static attitude. From a theoretical perspective, this approach maintains data confidentiality (the attacker cannot see sensor data) as well as data integrity and authenticity (it is not possible to fabricate/modification on sensor data without the right key). Nevertheless, from a practical perspective, it is necessary to analyze the effects of this solution on robot performance in order to avoid the inclusion of collateral undesired behaviors.

3.2. Materials

The robot used in the experiments was a mobile bi-manipulator named Karen, designed in our lab, and shown in **Figure 3**. The computation is carried out in a detachable Computational Unit (CU) powered by independent batteries, which provides easy scalability. Two different CUs were used in the experiments. CU-A is based on an Intel Core i7-3612QM CPU (four cores at 2.10 GHz) and 8 GB of RAM memory. CU-B uses an Intel(R) Atom(TM) CPUD525 (two cores at 1.80 GHz) with 2 GB of RAM memory. Karen has two main sensors: an Asus Xtion camera (640 × 480 pixels resolution, 30 fps) and a Hokuyo URG-04LX-UG01 laser sensor (683 points at 10 Hz).

The software running on these CUs is a combination of a GNU/Linux operating system, Trusty Tahr Ubuntu distribution (version 14.05.5) and ROS Indigo.[4]

3.3. Method

ROS defines different data types for each type of sensor. Besides, it is possible to generate user-defined custom messages. In this study, two types of messages were used: *LaserScan.msg* and *Image.msg*, both defined in the standard package *sensor_msgs*.[5] Laser messages have a fixed size (4,152 bytes). They include among other data the following information: scan angle, the angular distance between measurements, and range data. Image messages may have

[4]http://wiki.ros.org/indigo.
[5]http://wiki.ros.org/sensor_msgs.

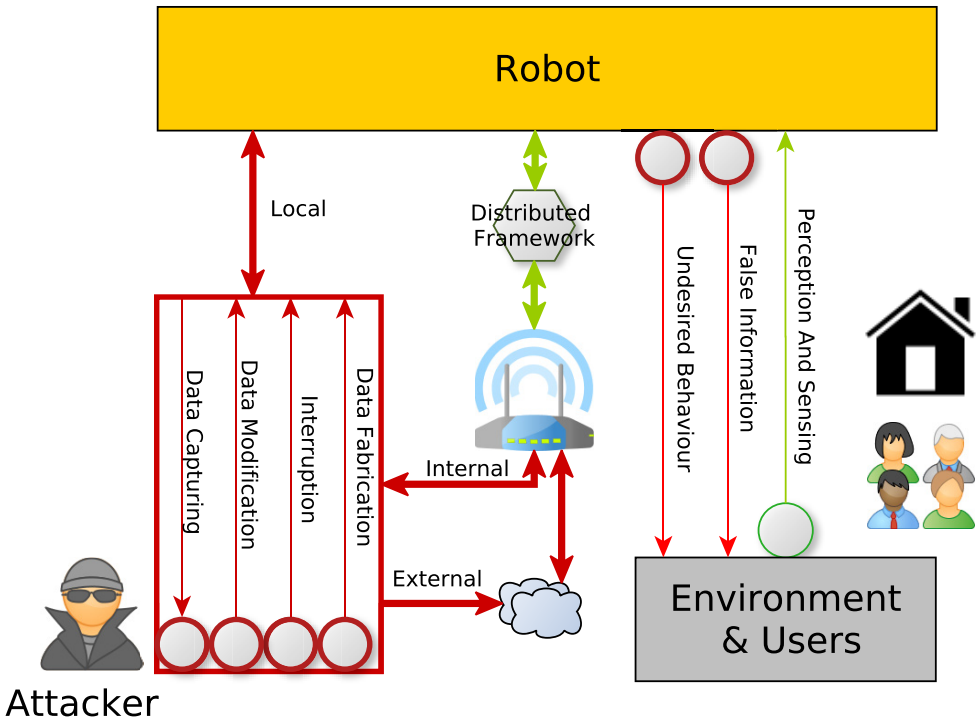

FIGURE 2 | A conceptual model of a cyber-attack to a robot.

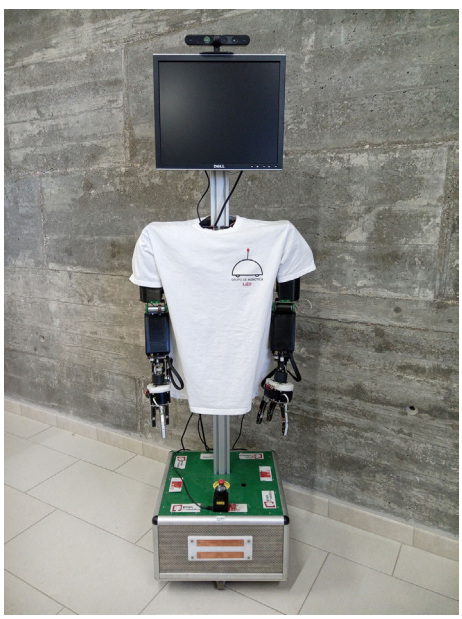

FIGURE 3 | Karen, the robot used in the experiment.

variable lengths. The average length of image messages sent in the experiments was 79,736.9 bytes, with a SD of 818.5 bytes. Image messages include information about image height and width, pixel encoding, and raw data with a height × width size.

As mentioned in section 1, three different symmetric encryption algorithms were used to encrypt the ROS messages: AES, 3DES, and Blowfish. Asymmetric encryption algorithms, such as RSA, are not used because the asymmetric encryption increases the size of the cryptogram and the symmetric encryption does not. In addition, the state of the art stated better CPU performance using symmetric encryption. In addition, it was necessary to select the mode of operation for the encryption algorithms (Dworkin, 2001). The encryption mode proposed by Iwata (2008) is a scheme for guaranteeing privacy and authenticity. There are two different modes of operation: a straightforward encryption, which is represented by the Electronic Code Book mode (ECB) and those modes whose encryption scheme is based on *Key IV*, represented by Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), XEX-based tweaked-codebook mode with ciphertext stealing XTS, and Counter (CTR) modes. These modes are used to guarantee the confidentiality but not message integrity.

In this study, the CBC[6] mode was chosen because of its high level of adoption and simpler implementation. The process for encrypting using CBC mode is based on combining plaintext blocks with the previous ciphertext blocks. The alternative is CTR mode, which is also widely used. CTR and CBC modes show similar encryption/decryption performances when using small

[6]For comparison reasons, we have used the same random SECRET KEY of 24 bytes (192 bits) in the three encryption algorithms and an initialization *Key IV* of 8 bytes (64 bits) for 3DES and Blowfish (PyCrypto does not allow longer keys) and a value of 16 bytes (128 bits) for AES.

blocks (Rogaway, 2011), ass well as in bigger blocks during the decryption process, due to their parallelization features.

In order to evaluate the effects of hardening on ROS, four elements are measured following the proposal evaluated in the state of the art section:

1. Network overhead: analyzed measuring the bytes introduced in the network flow.
2. CPU usage: overviews the total percentage of CPU in the robot and the percentage of CPU per process.
3. Autonomy effect: measures the power consumption and battery discharge rate.

Experiments were made using the same data using `rosbags`, that is, a file format in ROS for storing ROS message data. They are typically created by the `rosbag` command line tool, which subscribes to one or more ROS topics and stores the serialized message data in a rosbag file as it is received. A rosbag file is equivalent to a recording of the state of the robot for a period of time and can be used as a dataset. These `rosbag` files can be played back in ROS to the same topics they were recorded from, ensuring that the same conditions were applied in all the experiments.

Setup of the experiments was made consistently: 68 min of data were recorded using laser sensor and RGB camera captured on robot Karen and stored in a *rosbag* file. Same ROS nodes were launched in the same order in all the experiments. The same setup was repeated three times per each encryption algorithm and two computational units described above, CU-A and CU-B. **Figure 4** shows ROS setup running in each iteration. This configuration is presented in this way for testing reasons, in a realistic approach the laser and camera nodes are publishing encrypted from scratch.

Along with the setup, four benchmarking tools were used: *powertop*, *powerstat*, *dstat*, and ROS' statistics tools. These tools generate a set of files with the information about CPU performance, power consumption and messages sent and dropped. Bash scripts were used to process these files and Libre Office tools to generate figures shown in this paper.

Two constants were removed from the evaluation: ethernet power consumption and display consumption. We designed an experiment where the robot runs in autonomous mode and everything is computed on-board and where the display power consumption is not relevant in this context.

4. RESULTS

This section describes the results when the system plays the rosbag file and an encryption algorithm is applied. Data presented here shows the average of the three loops in our experimental setup. The section is organized, taking into account the different performance factors considered, such that subsection 4.1 shows the results regarding the throughput of ROS messages, subsection 4.2 analyzes the use of the CPU as proposed in Rihan et al. (2015), and subsection 4.3 details the power consumption as proposed by Elminaam et al. (2010).

4.1. Network Overhead

In order to get a full perspective of the throughput, we analyzed the bytes flooding the network when the system of encryption is active. Considering the total number of bytes transferred, AES delivers significantly more data than the other encrypting alternatives (1.5 MB by default, almost 5 MB when encryption is applied to the image messages). This volume can be reduced by applying ciphertext stealing techniques (Dieber et al., 2016), but this solution is not available in PyCrypto. **Figure 5** outlines the results, using as a reference 5 MB in the Y-axis. Left picture in **Figure 5** shows camera throughput and right picture presents laser throughput.

4.2. CPU Usage

CPU usage is the second parameter chosen for characterizing the performance of the alternatives. The same rosbag was used in the same environment and the *time* UNIX command was used to measure the load of the CPU.

Figure 6 shows the percentage of CPU used by different algorithms. In these, figures we have differentiated the encryption and decryption processes for each type of data because in some scenarios they may be running in different computing units.

Note that ROS is based on a "publish/subscription" paradigm. The encryption node "gets" a new message when it has finished a previous ciphering operation; if this operation takes more time than the generation of new data by the sensor, some messages may be skipped. ROS uses buffers of just one message for TCPROS communications. This explains why CU-B uses less than 100% of the CPU, while some messages are dropped.

4.3. Autonomy Effect

The third element analyzed was the autonomy effect on the robot "live" autonomy in terms of battery duration. This study proposes PowerTOP and GNU/Linux internal tools to gather this information in real time. PowerTOP is a tool provided by Intel to monitor processes (Benedict, 2012), it is able to estimate power consumption of each process running on the system. GNU/Linux provides the kernel tools (Power supply class) and *upower* command.

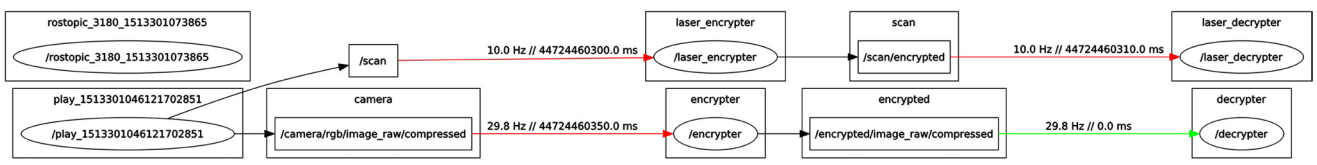

FIGURE 4 | ROS scenario for the experimental setup. From left to right: (1) rosbag player, (2) camera and laser topics, (3) encryption nodes, (4) topics from laser and camera with encrypted information, and (5) decryption nodes.

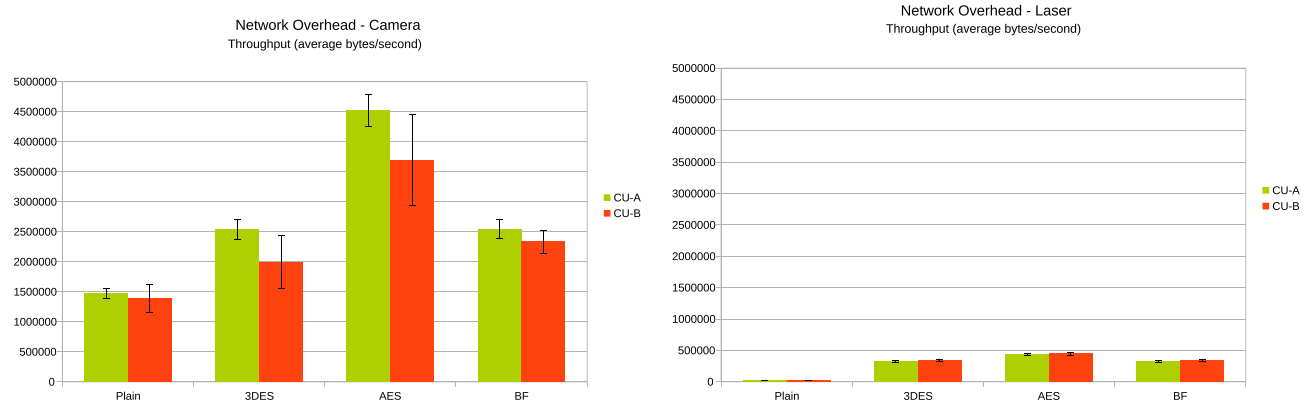

FIGURE 5 | Throughput in the three ciphered scenarios using CU-A and CU-B measured in bytes transferred to the net per second in average (camera and laser, respectively).

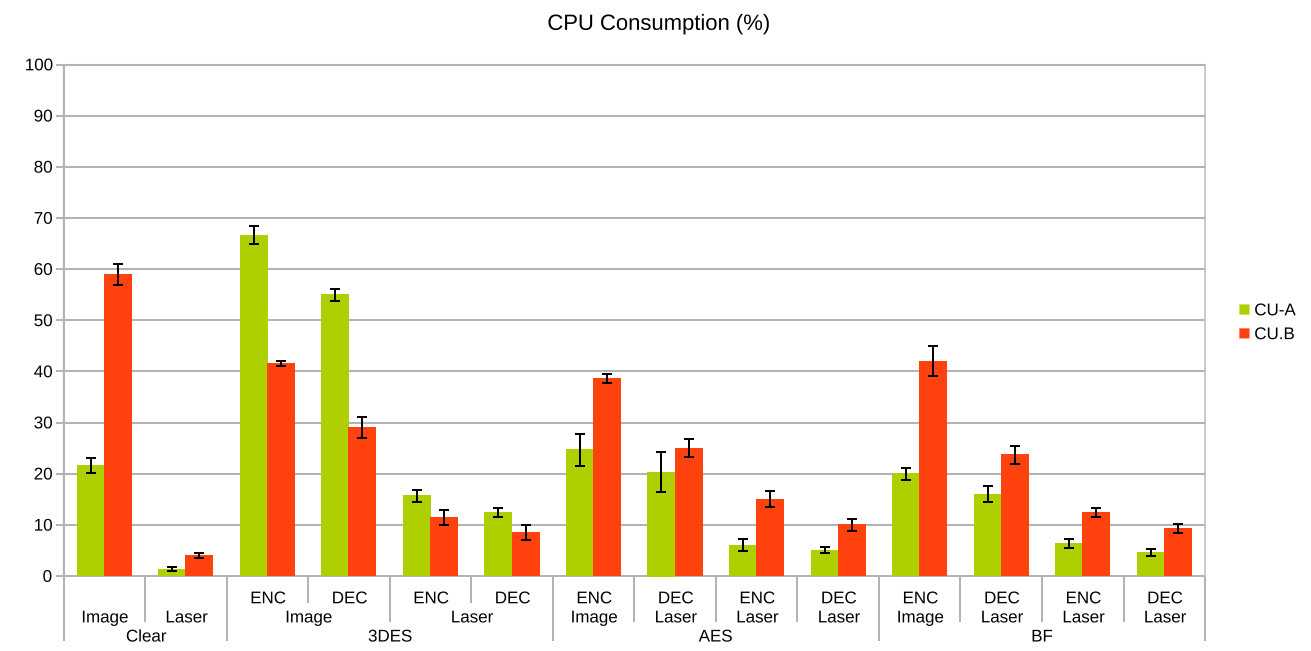

FIGURE 6 | Percentage of CPU used by different cryptographic algorithms using CU-A and CU-B.

Figure 7 shows total power used by each alternative (average and SD) for each of the CUs. PowerTOP was configured to dump the average power usage of the system (discharge rate) every 15 s.

Figure 8 shows the power usage differentiating the encryption and decryption processes (labeled as "ENC" and "DEC") for the different scenarios (labeled "L" for laser data and "I" for image) in the two CUs using the same methodology (PowerTOP measuring in mWatts every 15 s).

In order to evaluate the discharge rate of the whole robot, the C-rate metric has been used. A 1C-rate means that the discharge current will discharge the entire battery of the robot in 1 h. For a battery with a capacity of 100 Amp-h, this equates to a discharge current of 100 Amps.

The robot as a complete computing system presents different discharge rates in the different scenarios. The discharge rate is measured in three situations: the robot powering all devices, the robot running standard ROS, and the robot running hardened ROS. The C-rates on the CU-A, using a battery of 4,400 mAh and for the CU-B using a 5,200 mAh are presenting in **Table 1**.

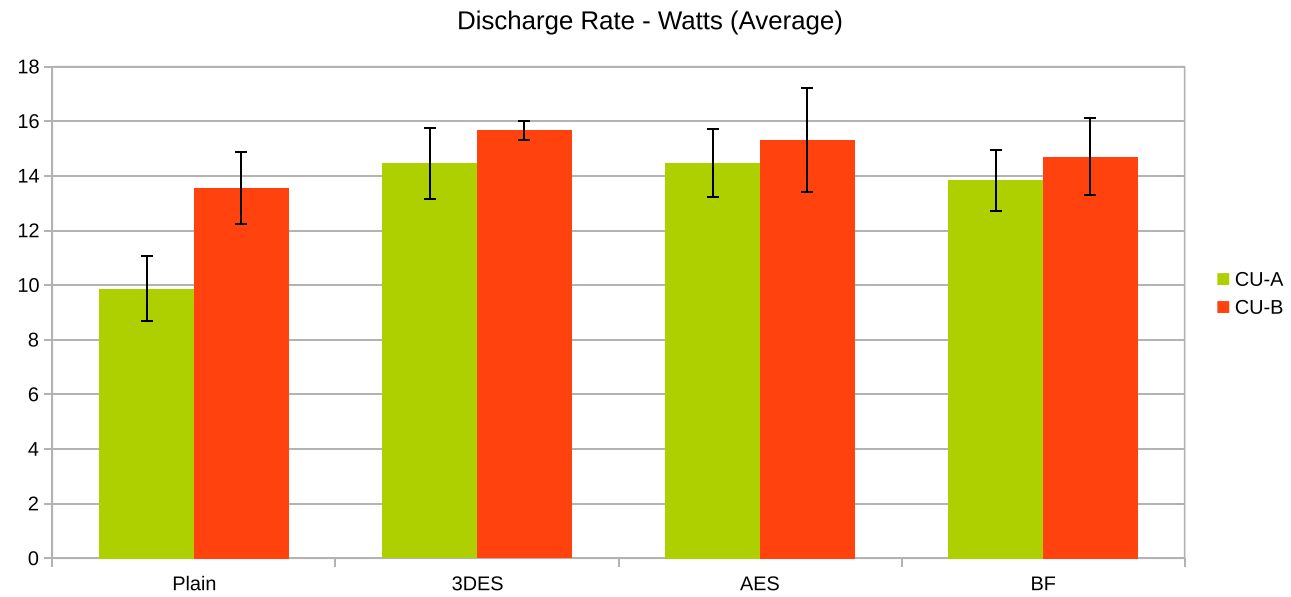

FIGURE 7 | Discharge rate in CU-A and CU-B when running normally (Plain) and when the different encryption solutions are running.

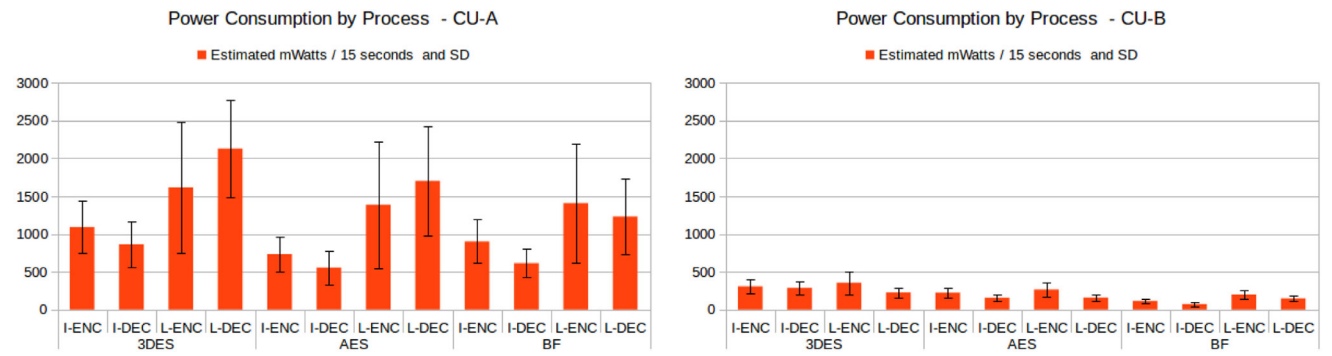

FIGURE 8 | Estimated power consumption defined by encryption/decryption process measured in mWatts in intervals of 15 s.

TABLE 1 | Discharge rate by CU (unit: C-rate, 1 C = 1 h of battery).

Battery	Switched-on mode	ROS mode	Hardening mode
4,400 mAh	0.28 C	0.37 C	0.45 C
5,200 mAh	0.2 C	0.3 C	0.4 C

5. DISCUSSION

This section reviews the effects of encrypting different blocks of information at the application level using an *ad hoc* solution. The proposal, based on ROS, PyCrypto, and python allows us to survey some critical scenarios to take into account before applying random solutions to an operating robot. The two computational units used in this research have been identified here as Powerful CU (i7) and Medium CU (Atom). Computational units such as Odroid or raspberry have not been tested to date but their theoretical computational capacity situates them below Medium CUs.

The results show AES to be a superior solution to apply, requiring less CPU than other encryption algorithms, but implying a large data overhead in the network, thus not well suited for multi-robot environments or applications that require continuous interaction with consoles or external devices. However, it is essential to emphasize that some modes of encryption are not an advantage from a security point of view. For instance, ECB-mode is not semantically secure and its use is discouraged. The encryption of two messages, which have two blocks of plain text in common, results in the same ciphered blocks applying the ECB mode. In addition, when it is used in images (cameras are one of the most extended sensors), it has well-known problems (Mehran and Khayyambashi, 2017).

This study presents different effects due to the encryption process. The results show that there is a correlation between the type of the message to cipher, the rate of messages and the capabilities of the computing platform when planning to use encrypted communications. Regarding the type of the messages, it is important to analyze the fields and their types. Experimental data show that big size chunks of data [image messages vs laser messages, long strings (Lera et al., 2016)] worsen the performance of the system in all alternatives. Thus, messages composed of short data chunks (small arrays, integers, string blocks string message lesser than 262,144 bytes) can be easily dealt with by all types of CUs usually available in robotic systems. This means that basic range sensors (laser, ultrasound, infrared etc.) can be quite easily hardened. With the focus on the rate of sensors (and message generation) sensors that generate data at a high rate (cameras, RGB-D sensors, LIDAR, etc.) suffer a big collateral effect: the rate of messages decreases and the CPU cost is higher. Finally, the characteristics of the CU will define the number of encryption/decryption operations it is possible to perform.

According to the experimental data gathered in our experiments and summarized in the previous section, we propose a taxonomy for the hardening of ROS communications according to the CUs and the type of messages involved in the application. We propose **Table 2** as a guide for robotic developers questioning the adequate hardening alternative for a particular sensorization of a robot and a given computing configuration. **Table 2** presents simple and complex messages as something that can be tolerated when the robot is assembling good CU. However, in the case of medium CU (for instance those models of Turtlebot using netbook computers, or the robot Nao), this task should be analyzed taking into account project needs and the public space where it is deployed. There are three different scenarios focused on message type: the ✓symbol indicates the data types that can be encrypted without performance problems in most robotic platforms; ↺ means that an *ad hoc* study should be carried out to analyze the particular sensor and CU; and ⇄is used to indicate that a trade-off should be found between security and robot performance because it might not be feasible to encrypt communications.

To illustrate this situation, it is necessary to think in a robot with multiple gathering sensors running at the same time. For instance, Pepper[7] robot, which is manufactured with a Medium CU, has two RGB cameras plus a depth sensor. In order to manage its multi-sensor features, it is needed to define the importance of ciphering, one, two, or the three sensors attending the final deployment scenario of the robot such as a bank or a mall. According to the individual performance by process of one single sensor (**Figure 8**), three camera sensors will increase the discharge rate and also the number of messages processed which will present a reduction of its regular performance. This situation hinders the decision-making process in real time given a performance under the usual frame rate of 10 Hz (Handa et al., 2012).

Experimental data gathered shows that both CUs are capable of managing the original ROS data flow. However, during this research, we observed that when dealing with different encryption modes, in particular, CBC mode, the performance of the CUs differs. On the one hand, under CTC mode, CU-A and CU-B maintains the original throughput for laser data when using the three algorithms, and also for image messages when AES and BF are used, but performance is reduced to 10% when using 3DES. On the other hand, CU-B working on CBC mode is only able to cope with laser messages. When dealing with the image sensor the ratio of processed messages is five out of every 30 (20%) when using 3DES; it is 15 out of every 30 messages (50%) when AES is used, and the system gets 11 out of every 30 messages (33%) for Blowfish algorithm.

At the same time, in a robotic scenario, it is a must to manage the robot autonomy to do tasks during long periods of time. Power-saving is a constraint to be taken into account when deciding the level of security. As discussed in the previous section, the discharge rate is affected by the algorithms. In our experiments, robot Karen has an independent CU, the battery provides a regular working flow of 3 h, but is reduced to 2:40 h when using BF, to 2:30 h when using AES and to 2:20 h when using 3DES. This scenario should be observed carefully by researchers or developers before applying one or the other. **Table 3** shows the estimated battery time in a given moment during the experiments (average). This information is gathered from the Linux Indicator Applet in each experiment 5 min after the experiment has started. At a practical level, this means that the encryption reduces (on average) almost half an hour the battery duration, and we are able to do barely two loops of the experiment before to recharge.

Regarding the use of the network, results show that encryption clearly imposes an overhead. When the communications between ROS nodes are performed using wired connections, the system is capable of dealing with it, even when heavy encryption algorithms are used, but if wireless communications are used between ROS nodes, it is necessary to consider communication issues. The experiments recreated a favorable scenario to analyze the throughput and no wireless communications were used.

In addition, it is also necessary to know the effects of encryption at the application level. In order to do so, the main robot

[7] www.softbankrobotics.com

TABLE 2 | Overview of encryption options data/CU: ✓: encryption should be applied; ↺: *AD HOC* encryption, i.e., sensor type and/or CU affects the encryption process, making it necessary to analyze the scenario; ⇄: it is not feasible to encrypt all communications in any circumstance.

Sensor data Types	Simple type messages			Complex messages			Custom messages		
CPU/Algorithm	3DES	AES	BF	3DES	AES	BF	3DES	AES	BF
Powerful CU	✓	✓	✓	✓	✓	✓	✓	✓	✓
Medium CU	✓	✓	✓	⇌	✓	✓	↺	✓	↺

TABLE 3 | Estimated battery duration presented in the Ubuntu Applet Indicator for the battery.

	Plain	3DES	AES	BF
CU A	3:00:00	2:20:00	2:33:00	2:40:00
CU B	3:13:00	2:15:00	2:28:00	2:34:00

operational modes were simplified to three: navigation, perception, and dialog, so that it is possible to find out cause–effect relationships in regular scenarios. **Table 4** outlines the effects over these three modes. In summary, it is totally affordable to cipher messages between nodes when Powerful CUs are available. But even in this situation, the 3DES method has higher computational requirements, and the three modes could be affected if particularly demanding algorithms were used, for instance, SLAM (Simultaneous Location And Mapping) or deep-learning algorithms. On the other hand, performance problems arise when using Medium CUs, even for basic navigation or perception modes. For instance, the ROS navigation stack showed problems running in Turtlebot when using an Atom processor.

The results obtained from the empirical experiments and the examples proposed in this discussion, allow us to generate a characterization model for cybersecurity in robots based on the encryption algorithm and the type of message (unit used for managing the information gathered by the sensors of the robot). The model is defined as follows: first, the researcher defines the security level needed following established methods: (1) following formal policies as those proposed by NIST, (2) following expired rules of security, (3) following non-formal policies, and (4) applying non-secure solutions. Second, researcher defines which are the requirements for the best robot performance, robot autonomy, and overhead added to the network protocols. Finally, it is necessary to establish the relationship between the first and the second step. This study proposes a solution customizing general ICT approaches as the one proposed by Liu et al. (2003) that presents a scenario of: actors, composed by users, attackers, and stakeholders; tasks, which should define a set of human and robot tasks performed on a given physical place; and the relationships (humans and/or robots). Second, the researcher identifies robot sensors and the messages associated with each sensor. The task will define the autonomy values, the net-overhead will be associated with the type of the communication method and the robot performance will be associated with the interaction level.

TABLE 4 | Effects of encryption over the classic operational systems of a robot.

		Encryption block		
CPU type	**Robot system**	**3DES**	**AES**	**BF**
Powerful CU	Navigation	⊙	✓	✓
	Perception	⊙	✓	✓
	Dialog	⊙	✓	✓
Medium CU	Navigation	▼	▼	▼
	Perception	⊙	⊙	⊙
	Dialog	⊙	⊙	⊙

✓ = *No problem*, ▼ = *Not affordable*, ⊙ = *Affordable with minor adjustments.*

Figure 9 presents our experimental approach associated with the model. Security refers to the strength of the encryption method according to NIST forecast: TDES will stand until 2013 (scored 2), AES 256 will be used further than 2030 (scored 3) and BF is not considered as standard (scored 1). A scale (from 0 to 3) was defined for the remaining three dimensions, where 0 signifies no effects on the robot performance, and 3 signifies the maximum impact on its performance. Values were assigned

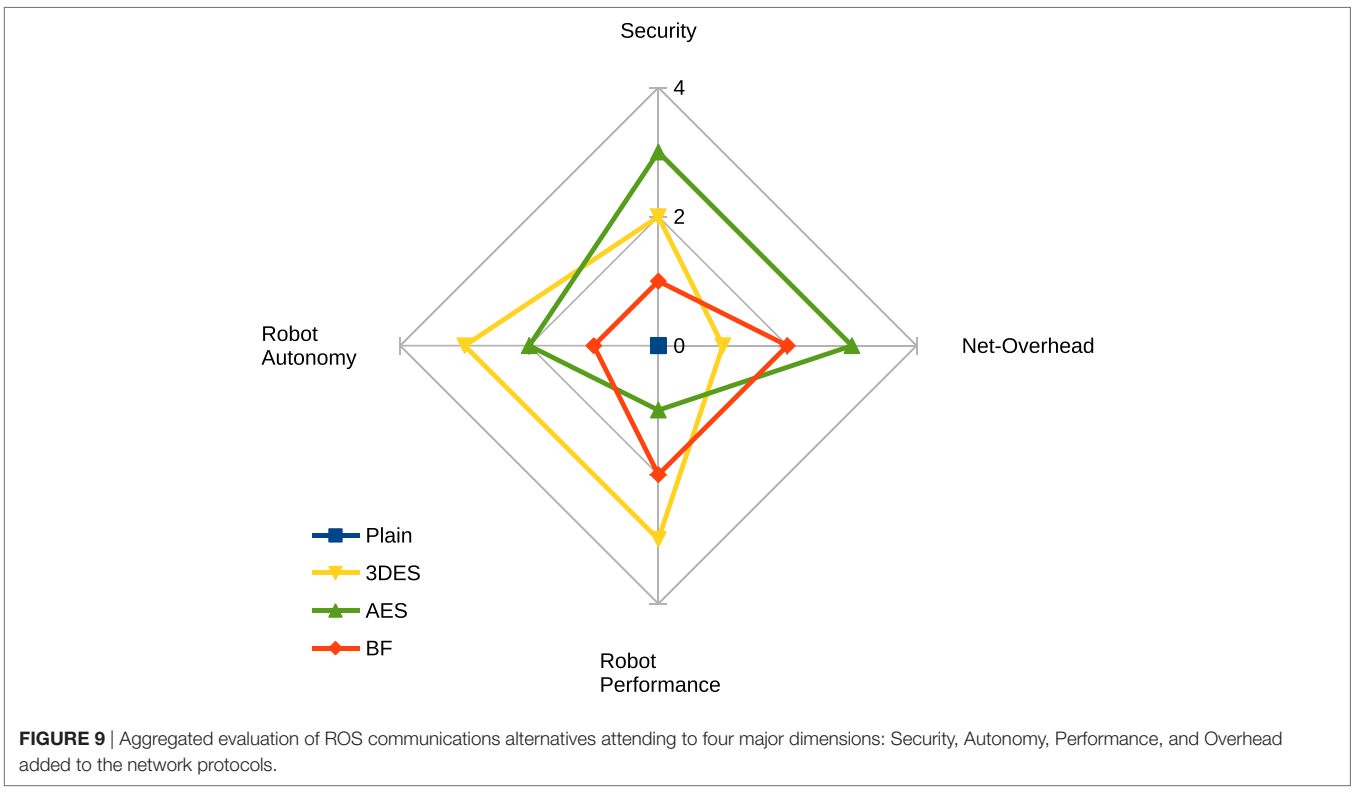

FIGURE 9 | Aggregated evaluation of ROS communications alternatives attending to four major dimensions: Security, Autonomy, Performance, and Overhead added to the network protocols.

according to previous discussions and define a robot using only two sensors.

6. CONCLUSION AND FURTHER WORK

In this article, we have identified the issues associated with the encryption process at the application level for hardening ROS. One major contribution described in this article is the characterization of three different hardening alternatives for the communication protocol (TCPROS) of ROS. We have empirically evaluated three different encryption algorithms for the most common sensors in two typical hardware configurations. **Figure 9** summarizes the results of the analysis made. Datasets used in the evaluation are available in our public `git` repository (it is necessary to clone): http://niebla.unileon.es/proyectos/publications/dataset-ros-security.git.

Another contribution derived from this empirical evaluation is the proposal of a taxonomy of robotic scenarios and the most convenient encryption algorithm to be used. **Table 2** summarizes this contribution. Our proposal does not require robotics developers to change existing software to accommodate our solution. Existing robotic applications can be hardened using our approach just adding standard encryption libraries.

AUTHOR CONTRIBUTIONS

FR-L and JB-C: software development, drafting state of the art in the field, contributions to the conception of the study and initial approach of the article. FR-L, JB-C, AG-H, CF-L, and VM-O: review, analysis, and interpretation of data gathered during the experiments; evaluation of the integrity of the data and research methods; FR-L, VM-O, and CF-L: review, corrections, and suggestions during the drafting process; and final approval of the version to be deliverable and published.

FUNDING

The authors would like to thank the next institutions for the partial financial support to this work: Spanish National Institute of CyberSecurity (INCIBE) under grant "Adenda21"; Junta de Castilla y León and FEDER funds under Research Grant No. LE028P17.

REFERENCES

Barker, E., and Barker, W. C. (2016). Cryptographic standards in the federal government. *NIST Spec. Publ.* 800, 175A.

Benedict, S. (2012). Energy-aware performance analysis methodologies for HPC architectures. An exploratory study. *J. Netw. Comput. Appl.* 35, 1709–1719. doi:10.1016/j.jnca.2012.08.003

Breiling, B., Dieber, B., and Schartner, P. (2017). "Secure communication for the robot operating system," in *2017 Annual IEEE International Systems Conference (SysCon), Montreal, Canada* (IEEE), 1–6.

Daemen, J., and Rijmen, V. (2013). *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer Science & Business Media.

Denning, T., Matuszek, C., Koscher, K., Smith, J. R., and Kohno, T. (2009). "A spotlight on security and privacy risks with future household robots: attacks and lessons," in *Proceedings of the 11th International Conference on Ubiquitous Computing* (Orlando, FL: ACM), 105–114.

Dieber, B., Kacianka, S., Rass, S., and Schartner, P. (2016). "Application-level security for ROS-based applications," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Daejeon: IEEE), 4477–4482.

Dworkin, M. (2001). *Recommendation for Block Cipher Modes of Operation*. Methods and techniques. National Institue of Standars and Techonology. Available at: www.dtic.mil/get-tr-doc/pdf?AD=ADA400014

Elminaam, D. S. A., Abdual-Kader, H. M., and Hadhoud, M. M. (2010). Evaluating the performance of symmetric encryption algorithms. *Int. J. Netw. Secur.* 10, 216–222.

Elminaam, D. S. A., Kader, H. M. A., and Hadhoud, M. M. (2008). Performance evaluation of symmetric encryption algorithms. *Int. J. Comput. Sci. Netw. Secur.* 8, 280–286.

Gueron, S. (2010). *Intel® Advanced Encryption Standard (AES) New Instructions Set*. Intel Corporation. Available at: http://www.pentium.md/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

Handa, A., Newcombe, R. A., Angeli, A., and Davison, A. J. (2012). "Real-time camera tracking: when is high frame-rate best?" in *European Conference on Computer Vision, Florence, Italy* (Springer), 222–235.

Hardjono, T., and Dondeti, L. R. (2005). *Security in Wireless LANS and MANS (Artech House Computer Security)*. Norwood, MA: Artech House, Inc.

Hood, D., and Woodall, W. (2016). "ROS 2 update – summary of alpha releases, architectural overview," in *ROSCon 2016, Seoul, Korea*. Available at: https://goo.gl/oCHR7H

Huang, J., Erdogan, C., and Zhang, Y. (2014). "ROSRV: runtime verification for robots," in *International Conference on Runtime Verification* (Springer), 247–254.

Iwata, T. (2008). "Authenticated encryption mode for beyond the birthday bound security," in *Progress in Cryptology – AFRICACRYPT 2008, Casablanca, Morocco* (Springer), 125–142.

Lee, G. S., and Thuraisingham, B. (2012). Cyberphysical systems security applied to telesurgical robotics. *Comput. Stand. Interfaces* 34, 225–229. doi:10.1016/j.csi.2011.09.001

Lera, F. J. R., Balsa, J., Casado, F., Fernández, C., Rico, F. M., and Matellán, V. (2016). *Cybersecurity in Autonomous Systems: Evaluating the Performance of Hardening ROS*, Málaga, Spain. 47.

Liu, L., Yu, E., and Mylopoulos, J. (2003). "Security and privacy requirements analysis within a social setting," in *Proceedings 11th IEEE International Requirements Engineering Conference, 2003, Los Alamitos, CA, USA* (IEEE), 151–161.

McClean, J., Stull, C., Farrar, C., and Mascareñas, D. (2013). "A preliminary cyber-physical security assessment of the robot operating system (ROS)," in *SPIE Defense, Security, and Sensing* (International Society for Optics and Photonics), 874110. doi:10.1117/12.2016189

Mehran, N., and Khayyambashi, M. R. (2017). Performance evaluation of authentication-encryption and confidentiality block cipher modes of operation on digital image. *Int. J. Comp. Netw. Infor. Secur.* 9, 30–37. doi:10.5815/ijcnis.2017.09.04

Morante, S., Victores, J. G., and Balaguer, C. (2015). Cryptobotics: why robots need cyber safety. *Front. Rob. AI* 2:23. doi:10.3389/frobt.2015.00023

Nadeem, A., and Javed, M. Y. (2005). "A performance comparison of data encryption algorithms," in *First International Conference on Information and Communication Technologies, 2005. ICICT 2005* (IEEE), 84–89.

Panchenko, A., and Pimenidis, L. (2006). "Towards practical attacker classification for risk analysis in anonymous communication," in *International Conference, Communications and Multimedia Security, Heraklion, Crete, Greece* (Springer Berlin Heidelberg), 240–251. doi:10.1007/11909033_22

Patil, P., Narayankar, P., Narayan, D., and Meena, S. (2016). A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish. *Procedia Comput. Sci.* 78, 617–624. doi:10.1016/j.procs.2016.02.108

Quigley, M., Conley, K., and Gerkey, B. (2009). "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, Vol. 3, (Kobe, Japan), 5–10.

Raymond, J.-F. (2001). "Traffic analysis: protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies* (Springer Berlin Heidelberg), 10–29. doi:10.1007/3-540-44702-4_2

Rihan, S. D., Khalid, A., and Osman, S. E. F. (2015). A performance comparison of encryption algorithms AES and DES. *Int. J. Eng. Res. Technol.* 4, 151–154.

Rogaway, P. (2011). "Evaluation of some blockcipher modes of operation," in *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*.

ROS Wiki. (2014). *ROS Technical Overview*. Available at: http://wiki.ros.org/ROS/Technical%20Overview

Sabaliauskaite, G., Ng, G. S., Ruths, J., and Mathur, A. P. (2016). "Empirical assessment of methods to detect cyber attacks on a robot," in *IEEE 17th International Symposium on High Assurance Systems Engineering (HASE), Orlando, Florida* (IEEE), 248–251.

Sattarova Feruza, Y., and Kim, T.-H. (2007). It security review: privacy, protection, access control, assurance and system security. *Int. J. Multimedia Ubiquitous Eng.* 2, 17–32.

Schneier, B. (1993). "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in *International Workshop on Fast Software Encryption* (Cambridge: Springer), 191–204.

Schneier, B., Hall, C., and Ferguson, N. (1999). "Performance comparison of the {AES} submissions," in *The Second AES Candidate Conference; National Institute of Standards and Technology, Gaithersburg, MD*, 15–34.

Smid, M. E., and Branstad, D. K. (1988). Data encryption standard: past and future. *Proc. IEEE* 76, 550–559. doi:10.1109/5.4441

Vuong, T. P., Loukas, G., and Gan, D. (2015). "Performance evaluation of cyber-physical intrusion detection on a robotic vehicle," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)* (Liverpool: IEEE), 2106–2113.

White, R., Christensen, Henrik, I., and Quigley, M. (2016). SROS: securing ROS over the wire, in the graph, and through the kernel. *CoRR*. abs/1611.07060. Available at: http://arxiv.org/abs/1611.07060

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

The reviewer, HK, and handling editor declared their shared affiliation.