CryptoROS: A Secure Communication Architecture for ROS-Based Applications

Roham Amini[1*], Rossilawati Sulaiman[2], Abdul Hadi Abd Rahman Kurais[3**]
Faculty of Information Science & Technology
Universiti Kebangsaan Malaysia
Bangi Selangor, MALAYSIA

***Abstract*—Cyber-attacks are a growing threat to future robots. The shift towards automatization has increased relevance and reliance on robots. Securing robots has been secondary or ternary priority and thus robots are vulnerable to cyber-attacks. Securing robots must become an essential (built-in) part of the design rather than being considered as a subsequent (later) add-on. ROS is a widely used and popular open source framework and robots using ROS are increasing in popularity. However, ROS is vulnerable to cyber-attacks. ROS needs to be secured before robots using ROS reach mass market. This study aims at proposing an architecture to secure ROS, using cryptography mechanism, which addresses the most common ROS safety issues. The advantages of our proposed secure architecture, CryptoROS, is that no changes to ROS software libraries and tools is required, it works with all ROS client libraries (e.g. rospy, roscpp) and rebuilding nodes is not necessary.**

Keywords—Robotics; ROS; cyber security; cryptography; access control

I. Introduction

Autonomous robots are expanding not only in science-fiction movies, but in our regular, everyday tangible world. For example, applications of robots are used in education [1], [2], accounting [3], target searching and detection [4], [5], and many more. With robots becoming further ubiquitous in society, cyber-attacks are rapidly growing into a cogent issue. Home service robots, autonomous vehicles, industrial automation, along with many other robotics domains offer a route for the spread of cyber threats into real-world risks. Personal robots with the potential to integrate with the Internet of Things (IoT) may very well be targeted, in the same fashion as PCs and smartphones, and lead to violations of privacy and breaches of confidentiality. For robot software development, ROS, a group of open source software libraries and tools, is used. Programmable robots are becoming increasingly popular, and as robots appear more within the society, the safety of ROS is becoming an important concern and should be considered vital because it may become a target for breaches of confidentiality and / or violation of integrity [6]-[8].

In ROS, every node running has a XML-RPC URI. XML-RPC is a remote procedure call, encoding complex data structures using XML and transmitting / transporting them using HTTP [9]. As depicted in figure 1, the publisher advertises, via the master's XML-RPC, its intent to publish to topic chatter. Then the subscriber subscribes to topic chatter via the master's XML-RPC. In response the master returns the publisher's XML-RPC URI to the subscriber. The subscriber then requests and negotiates a topic connection via the publisher's XML-RPC. In response the publisher returns the proper setting for the selected topic transport to the subscriber. Using the provided setting, the subscriber then establishes a new connection to the publisher [10].

The remainder of this paper has been structured as follows: next section explores the related work. Section three introduces the proposed architecture used to secure ROS. Finally, section four discusses the ROS issues fixed.

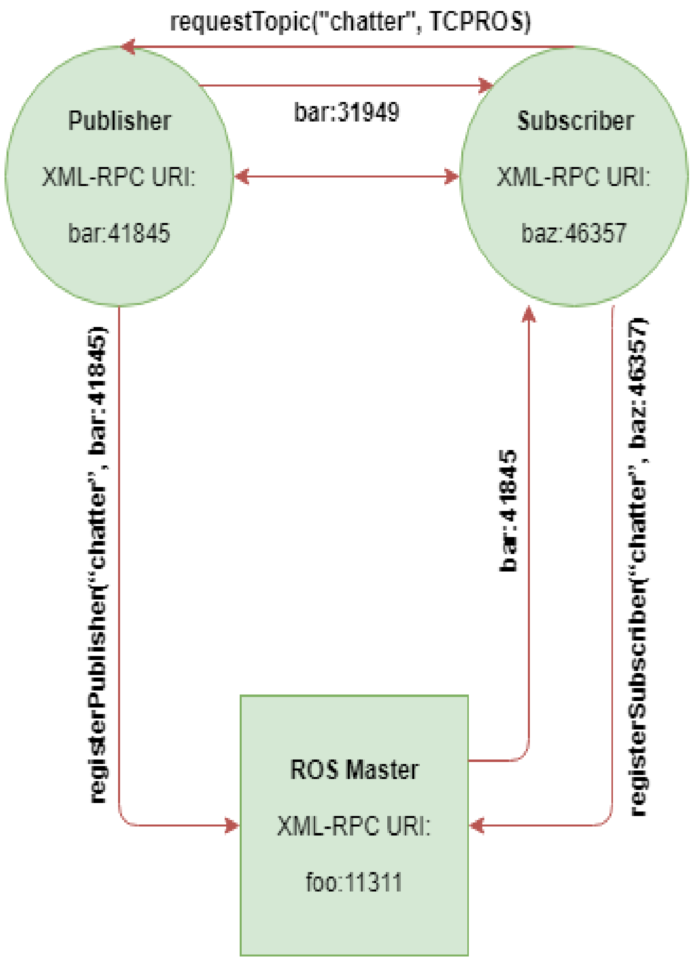

requestTopic("chatter", TCPROS)
Publisher
XML-RPC URI:
bar:41845
bar:31949
Subscriber
XML-RPC URI:
baz:46357
registerPublisher("chatter", bar:41845)
bar:41845
registerSubscriber("chatter", baz:46357)
ROS Master
XML-RPC URI:
foo:11311

Fig. 1. ROS Architecture.

* 1st corresponding author, ** 2nd corresponding author.

II. Background Study

The development of ROS is influenced by features and properties most valued by robotics researchers and therefore fails to provide any protection against cyber-attacks. For example, a newly created node replaces an existing node with the same name, because nodes need to be named uniquely. Therefore, an attacker can shut down / kill a node and replace it, simply by running a node with the same name as the target node [6], [11]. A node can freely publish messages to a random / chosen topic without prior authorization. An unauthorized node publishing malicious messages to a topic can cause an unforeseen motion by a robot that damages its surroundings and / or harms nearby humans. A node can freely and without prior authorization subscribe to a random / chosen topic and receive all messages published to this topic. These messages might contain confidential information. A node can freely publish large number of messages to a random / chosen topic, preventing the subscriber of this topic from carrying out meaningful information processing and causing a denial of service. The topic transport channel is not secure. It reveals messages to unauthorized persons (breach of confidentiality), cannot detect unauthorized intentional or unintentional alteration of messages (violation of integrity), cannot prove that the involved parties (e.g. publishers, subscribers) are who they say they are [7], [12].

An attacker with expertise in ROS can execute a man-in-the-middle attack by acting as a publisher to a subscriber and as a subscriber to a publisher. An attacker with adequate knowledge and background in cyber security can find the XML-RPC URI of the master and armed with this information [12]:

1) The attacker calls the remote procedure “getSystemState” at the master and retrieves a list representing the names of current publishers, subscribers and service providers.

2) The attacker calls the remote procedure “lookupNode” at the master, provides the name associated with the targeted publisher / subscriber as parameter and retrieves the XML-RPC URI of that publisher / subscriber.

3) The attacker calls the remote procedure “publisherUpdate” at the subscriber and provides, among other parameters, the XML-RPC URI of the publisher under his / her control.

4) The attacker executes a man-in-the-middle attack and intercepts, monitors (passive attack), if desired alters / changes (active attack) and reroutes the conversation as shown in figure 2.

During DEF CON 20 conference [13], a car-like robot equipped with two cameras, a compass and a single board computer running Linux and ROS was deployed to emulate and experience the cyber-physical issues related to mobile robots built using ROS. Attendees interacted (e.g. drive the robot) with the physical robot via a webpage and were asked to exploit the vulnerable mobile robot. During the conference, an attendee with knowledge and background in ROS injected / published malicious messages and operated the robot without interacting with the webpage.

There are many researches that have been done to address ROS safety issues. ROSRV was introduced in [11], which has been designed in such a way that no changes to ROS software libraries and tools is required. ROSRV intercepts all requests to master and monitors and if required alters the messages, thus enforcing access control policies and monitoring safety properties. However, ROSRV transmits unencrypted traffic, disclosing private data and failing to prevent unauthorized alteration of data. ROSRV also relies and enforces access control policies based on the source IP address of the request. This exposes the architecture to IP spoofing. ROSRV could encounter scaling problems because all the monitors reside in the same multithreaded process.

Transport Layer Security (TLS) was used in SROS [6]. SROS encrypts all network traffic using TLS by changing the ROS client library, or more specifically rospy client library. Each node is supplied a X.509 certificate, with the access control policies embedded within the X.509 certificate extensions. One drawback is that at the time of writing this paper SROS only supported rospy client library with TCPROS. Another drawback is that because the access control policies are embedded within the X.509 certificate extensions, 1) mutating a node’s permissions requires revocation of the current X.509 certificate and issuance of a new one, 2) the access control policies are made public.

In [7] the authors described a concept similar to SROS but implemented by changing the roscpp client library. However, it fails to secure the master and the request / response sent / received via XML-RPC.

A scheme was introduced by [8] where by a node publishes messages in clear to a topic (/sensor/messages). An encrypting node subscribes to this topic (/sensor/messages), performs message encryption and publishes it to another topic (/sensor/encrypt/messages). A decrypting node subscribes to this topic (/sensor/encrypt/messages) and performs message decryption. The symmetric key is stored within the master and is only known by authorized entities. However, this testbed fails to prevent nodes from subscribing to /sensor/messages topic, which results in exposure of messages to unauthorized entities, a clear breach of confidentiality. It also fails to detect intentional or accidental alteration of messages, a clear violation of integrity. The testbed does not also check that the involved entities are who they say they are (no authenticity) [12].

In [14] the authors introduced an architecture in which prior to publishing and / or subscribing to a topic, nodes send their login credentials to an authorization node with the help of a set of overloaded / overridden functions. The authorization node generates a special key and returns it to the node to be included in all future conversations. however, the authors have mentioned that ROS uses SSH to secure all conversations. This is not true and will allow attackers to easily break / bypass their scheme. They also require changes to be made to ROS client libraries.

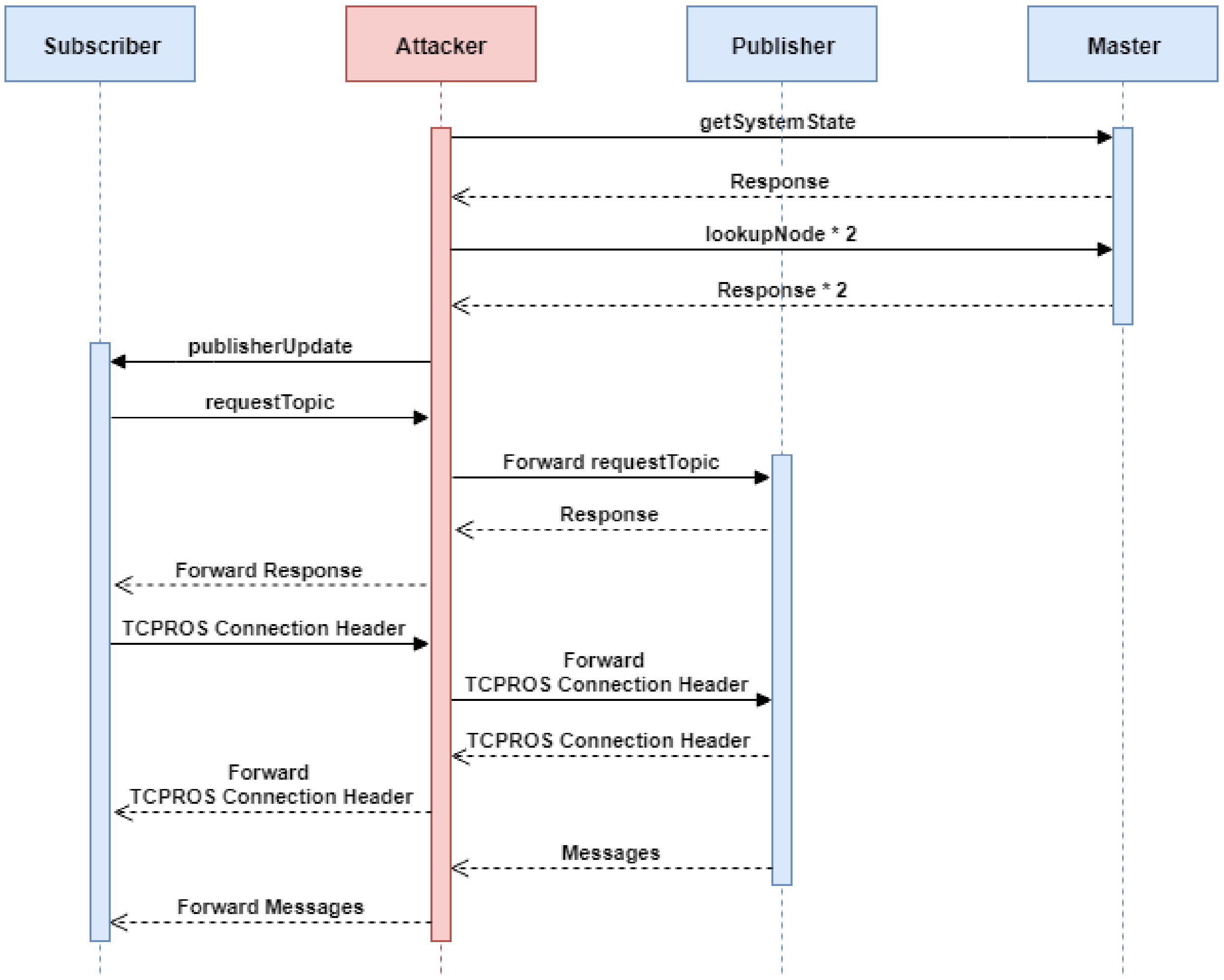


Fig. 2. Man-in-the-Middle Attack.

III. Proposed Architecture

A. Security Requirements

In our proposed architecture, we focus on the peer-to-peer conversations between nodes, which have to be confidential and checked for integrity violation. The Computation Graph [15] must be available and functional at all times. This involves decreasing the attack surface for denial of service attacks. The involved entities must also be scrutinized to ensure they are who they say they are. Nodes should not be allowed to publish / subscribe to a topic or advertise / call a service without prior authorization.

B. Proposed Secure ROS Architecture: CryptoROS

CryptoROS has been designed to fix some of the safety issues related to ROS. Manager as the name implies manages all nodes running on a computer. The Authorization Server checks the Manager's credentials and creates an Access Token representing the predefined set of actions the Manager has been authorized to perform as shown in figure 3. The Manager and the Authorization Server, each has been issued a X.509 certificate by a CA and supplied / configured with all the intermediate CA certificates to chain to the root CA certificate.

The entire conversation is secured using TLS 1.2, therefore the Access Token is never made public. The Access Token is made up of three parts: header, payload, and signature. The payload contains, among other claims, an expiry date claim. Involved parties perform a signature check to ensure the information contained in the Access Token has not been altered / changed (integrity check) and the Access Token has been created by a trusted entity (authenticity check). Therefore, the Authorization Server has been configured to use the private key associated with one of the intermediate CA certificates to create the signature.

As shown in figure 4, unbeknown to the Publisher / Subscriber, Publisher / Subscriber calls a remote procedure at the Manager and announces its intention to publish / subscribe to topic chatter (step 1 and 4). The Manager generates an Interceptor (step 2 and 5). The Interceptor announces, via the ROS Master's XML-RPC, its intent to publish / subscribe to topic chatter (step 3 and 6). In response to step 6, the ROS Master returns the Interceptor_P's XML-RPC URI to the Interceptor_S (step 7). Henceforth, the Interceptor acts as a publisher to a subscriber (step 8) or as a subscriber to a publisher, transparently intercepting, monitoring and if required altering / changing the conversations between Publisher and Subscriber.

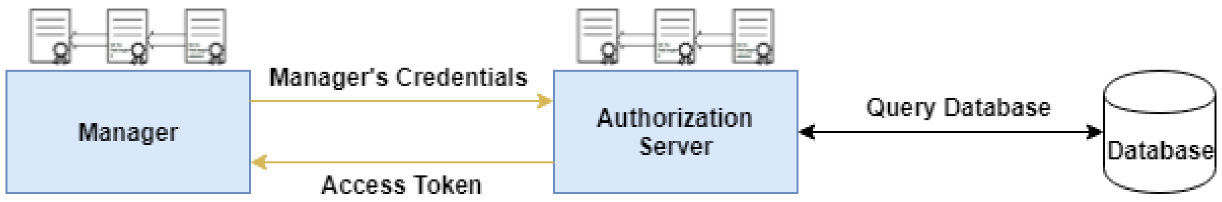


Fig. 3. Requesting an Access Token.

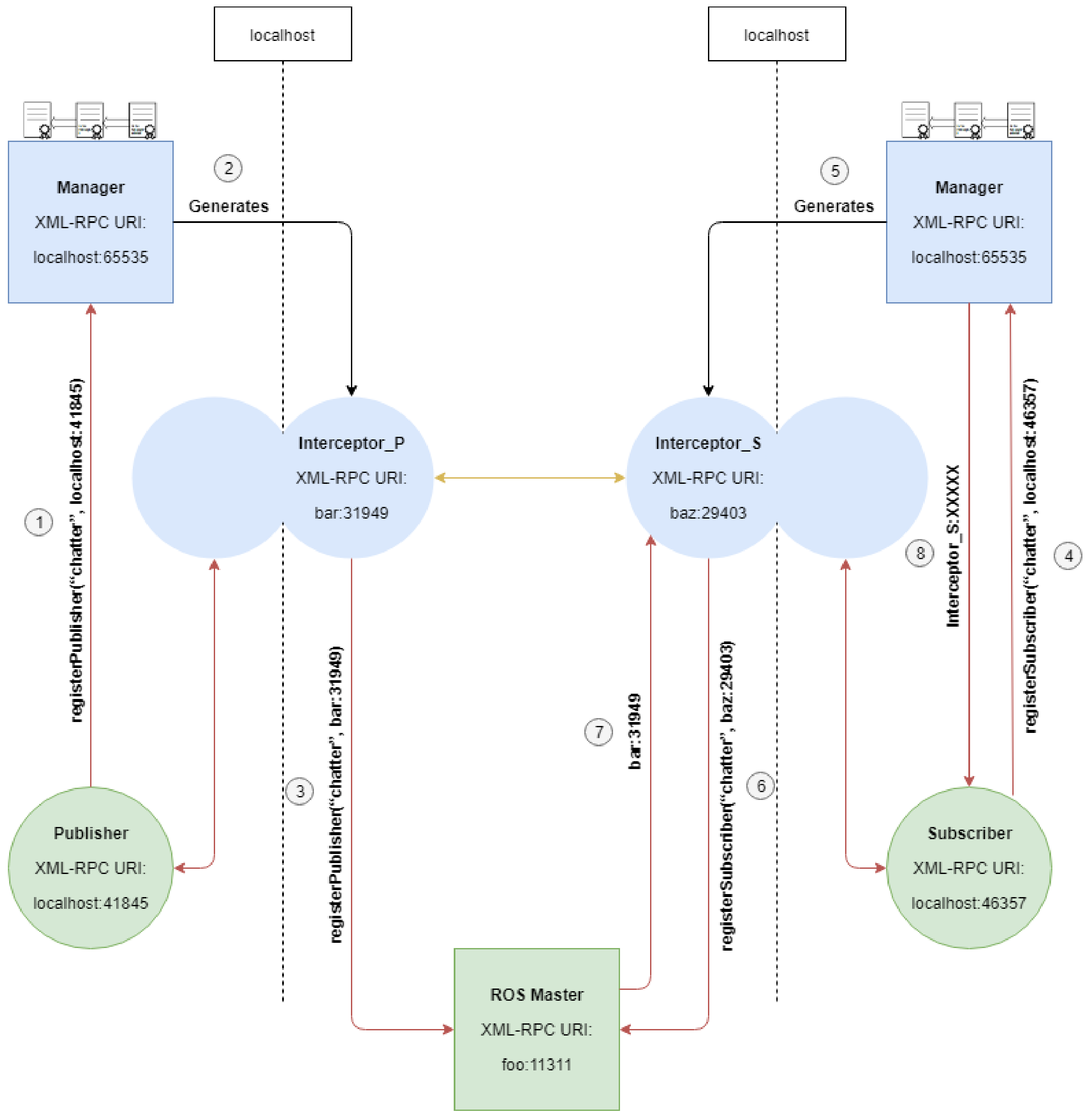


Fig. 4. CryptoROS Architecture. Orange Arrow Means the Conversation has been Secured using TLS 1.2.

In summary, the Publisher and the Subscriber have been configured to contact the Manager instead of the ROS Master by setting the ROS_MASTER_URI environment variable to the IP address and port number of the Manager. The Manager generates an Interceptor for each node. An Interceptor intercepts, monitors and if required alters all network traffic to / from the node and decrypts / encrypts them accordingly. The Interceptors act as publishers to subscribers and as subscribers to publishers. All conversations (network traffic) between the Interceptors are secured using TLS 1.2 except the XML-RPC request / response sent / received.

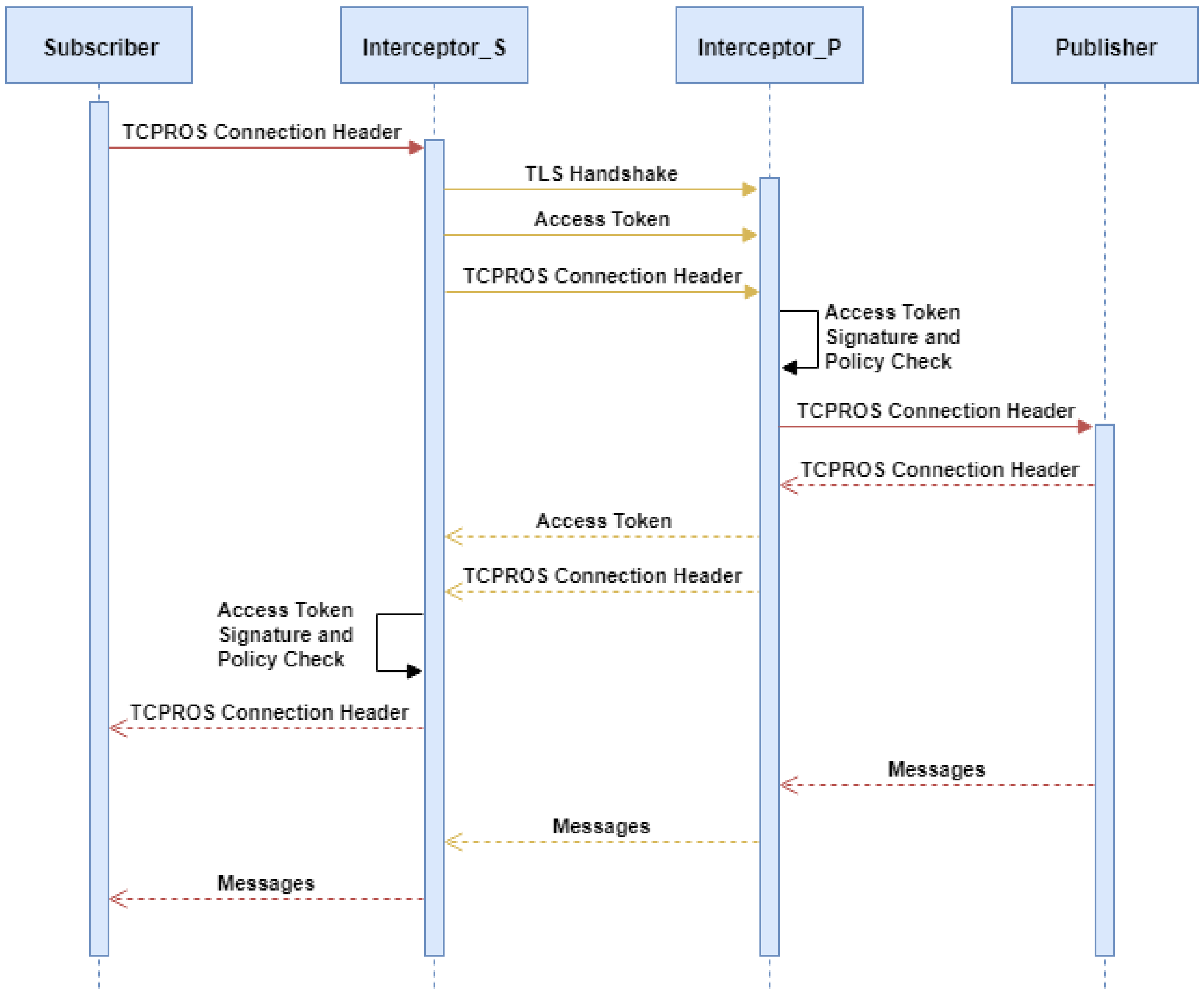


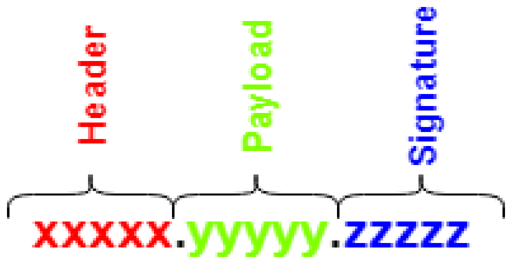
Fig. 5. Sequence Diagram. During the TLS Handshake, an Interceptor uses the X.509 Certificate Belonging to the Manager that Generated it. Orange Arrow Means the Conversation has been Secured using TLS 1.2.

As depicted in figure 5, upon receiving an Access Token, the Interceptor performs a signature check and then inspects the access control policies embedded within the Access Token to ensure the node is allowed to publish / subscribe to a topic or advertise / call a service. The expiry date of the Access Token is also checked. The same process applies to services and service clients.

The manager and the nodes (e.g. Publisher, Subscriber) have been bound to 127.0.0.1. This is done to prevent remote machine connections.

C. Access Token

As mentioned above the Access Token is made up of three parts: header, payload, and signature. The header contains the algorithm claim which denotes the cryptographic / signing algorithm used (e.g. RSASSA PKCS1 v1.5 using SHA-256). The payload contains, along with the access control policies, an issuer, a subject, and an expiry date claim. The issuer holds the unique / distinguished name of the entity that issued the Access Token. The subject contains the unique / distinguished name of the party that this Access Token bears claims about. The expiry date holds the date and time after which this Access Token is no longer considered usable. Figure 6 shows the Access Token structure.



6. Access Token.

TABLE I. COMPARISON BETWEEN ARCHITECTURES

Scheme	Advantage	Disadvantage
CryptoROS	- secures / encrypts peer-to-peer conversations between nodes. - stops unauthorized publishers, subscribers, services and service clients. - reduces the attack surface for DoS attacks. - no changes to ROS software libraries and tools. - supports all ROS client libraries.	- unsecured / unencrypted XML-RPC requests / responses sent / received. - does not protect ROS Master.
ROSRV	- no changes to ROS software libraries and tools.	- unsecured / unencrypted network traffic and reliance on IP addresses when enforcing access control policies exposes the architecture to a wide variety of attacks.
SROS	- secures / encrypts all network traffic. - stops unauthorized publishers, subscribers, services and service clients. - notably reduces the attack surface for DoS attacks.	- changes ROS software libraries and tools. - supports rospy only.

Table I compares CryptoROS with some of the solutions discussed previously and list some of their advantages and disadvantages.

IV. CONCLUSIONS AND FUTURE WORK

CryptoROS has been designed in such a way that no changes to ROS software libraries and tools is required. Additionally, rebuilding nodes is not required in order to benefit from the secure conversation channel. CryptoROS also works with all ROS client libraries regardless of the programming language they have been implemented / written in.

With our approach we managed to prevent unauthorized publishing and subscribing because the TLS handshake for the inbound and the outbound peer-to-peer connection will fail, prohibiting / preventing malicious nodes which are not supposed to be part of a specific conversation from injecting and / or eavesdropping data. The attack surface for denial of service in ROS has also been decreased. The Interceptors could be configured to drop XML-RPC shutdown requests, preventing attackers from shutting down / killing nodes.

With this approach we also made sure the messages and the service requests / responses will not be disclosed to unauthorized persons (confidentiality), any unauthorized intentional or accidental alteration of them will be detected (integrity) and we also made sure the involved entities are who they say they are (authenticity).

Some deployed robots might have inadequate computational power. Therefore, as a future work we will implement the proposed secure architecture and measure the performance impact on both the CPU and network traffic. In addition, we will attempt to secure master and XML-RPC requests / responses sent / received.

ACKNOWLEDGMENT

This research was supported by Universiti Kebangsaan Malaysia internal grant (Reference code: GGPM-2017-040).

REFERENCES

[1] N. F. A. Zainal, R. Din, M. F. Nasrudin, S. Abdullah, A. H. A. Rahman, S. N. H. S. Abdullah, K. A. Z. Ariffin, S. M. Jaafar, and N. A. A. Majid, “Robotic prototype and module specification for increasing the interest of Malaysian students in STEM education,” International Journal of Engineering and Technology, vol. 7, no. 3.25, pp. 286-290, Jan, 2018.

[2] L. P. E. Toh, A. Causo, P. W. Tzuo, I. M. Chen, and S. H. Yeo, “A Review on the Use of Robots in Education and Young Children,” Educational Technology and Society, vol. 19, no. 2, pp. 148-163, 2016.

[3] D. Fernandez and A. Aman, “Impacts of Robotic Process Automation on Global Accounting Services,” Asian Journal of Accounting and Governance, vol. 9, pp. 141-150, 2018.

[4] B. Nakisa, M. N. Rastgoo, M. Z. A. Nazri, and M. J. Nordin, “Target searching in unknown environment of multi-robot system using a hybrid particle swarm optimization,” Journal of Theoretical and Applied Information Technology, vol. 96, no.13, pp. 4055-4065, July, 2018.

[5] A. H. A. Rahman, K. A. Z. Ariffin, N. S. Sani, and H. Zamzuri, “Pedestrian Detection using Triple Laser Range Finders,” International Journal of Electrical and Computer Engineering (IJECE), vol. 7, no. 6, pp. 3037-3045, Dec, 2017.

[6] R. White, H. I. Christensen, and M. Quigley, “SROS: Securing ROS over the wire, in the graph, and through the kernel,” IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS), 2016.

[7] B. Breiling, B. Dieber and P. Schartner, "Secure communication for the robot operating system," 2017 Annual IEEE International Systems Conference (SysCon), Montreal, QC, 2017, pp. 1-6.

[8] F. J. R. Lera, J. Balsa, F. Casado, C. Fernandez, F. M. Rico, and V. Matellan, “Cybersecurity in Autonomous Systems: Evaluating the performance of hardening ROS,” Proc. XVII Workshop of Physical Agents, Spain, Málaga, 2016, pp. 47-53.

[9] XML-RPC.Com. (1999, June 14). Retrieved September 9, 2018, from http://xmlrpc.scripting.com/

[10] Vilches, V. M. (Ed.). (2014, June 15). ROS Technical Overview. Retrieved September 9, 2018, from http://wiki.ros.org/ROS/Technical Overview

[11] J. Huang, C. Erdogan, Y. Zhang, B. Moore, Q. Luo, A. Sundaresan, and G. Rosu, “ROSRV: Runtime Verification for Robots,” International Conference on Runtime Verification, 2014, pp. 247-254.

[12] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, “Security for the Robot Operating System,” Robotics and Autonomous Systems, vol. 98, no. C, pp. 192-203, Dec, 2017. doi: https://doi.org/10.1016/j.robot.2017.09.017

[13] J. McClean, C. Stull, C. Farrar, and D. Mascarenas, “A preliminary cyber-physical security assessment of the Robot Operating System (ROS),” SPIE Defense Security and Sensing, Baltimore, Maryland, United States, 2013.

[14] R. Dóczi et al., "Increasing ROS 1.x communication security for medical surgery robot," 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, 2016, pp. 004444-004449.

[15] Romero, A. M. (Ed.). (2014, June 21). ROS Concepts. Retrieved September 9, 2018, from http://wiki.ros.org/ROS/Concepts