Evaluation of Peer Robot Communications using CryptoROS

Abdul Hadi Abd Rahman[1*], Rossilawati Sulaiman[2†], Nor Samsiah Sani[3*], Afzan Adam[4*], Roham Amini[5‡]
*Center for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia
†Center for Cyber Security, Universiti Kebangsaan Malaysia
‡Faculty of Information Science and Technology, University Kebangsaan Malaysia

***Abstract*—The demand of cloud robotics makes data encryption essential for peer robot communications. Certain types of data such as odometry, action controller and perception data need to be secured to prevent attacks. However, the introduction of data encryption caused increment of overhead for data stream communication. This paper presents an evaluation of CryptoROS architecture on Robot Operating System (ROS) which focused on peer-to-peer conversations between nodes with confidentiality and integrity violation. OpenSSL is used to create a private key and generate a Certificate Signing Request (CSR) that contains public key and a signature. The CSR is submitted to a Certificate Authority (CA) to chain the root CA certificate and encryption of RSA private key with AES-256 and a passphrase. The protected private key are securely backed up, transported, and stored. Experiments were carried out multiple times with and without the proposed protocol intervention to assess the performance impact of the Manager. The results for different number of messages transmitted each time increased from 100, 250 to 500 with performance impact 1.7%, 0.5% and 0.2%, respectively. It is concluded that CryptoROS capable of protecting messages and service requests from unauthorized intentional alteration with authenticity verification in all components.**

***Keywords*—Robot communication; encryption; robot operating system**

I. Introduction

In robot applications, robot sensors collect huge amount of data from the environment to characterize the situation. Certain types of data require encryption due to prevent attacks such as odometry, action controller and perception data. There are various types of data such as strings, images and point cloud data [1], [2], [3]. Robotic Operating System (ROS) provides a reliable platform for robot application but is vulnerable to cyber-attacks. It needs to be embedded with security function before obots using ROS platform reach mass market. Despite the clear advantages of robot intergration in ROS, it lacks any security protection, which makes robots insecure and prone to malicious attacks especially in robot communications [4], [5], [6].

With the demand of cloud robotics in connecting multiple robots, the encryption of data communication over the network is essential. At current state, a node in ROS can freely publish messages to a random/chosen topic without prior authorization. It can freely and without prior authorization subscribe to the topic and receive all the messages. A node can freely publish large number of messages, preventing the subscriber of this topic from carrying out meaningful information processing and causing a denial of service [7], [8]. The topic transport channel is not secure. It reveals messages to unauthorized persons, unable to detect unauthorized intentional or unintentional alteration of messages, and also cannot prove that the involved parties [9]. Therefore, there is a need to provide a total solution to overcome these situations.

Previous researches introduced various encryption solutions for robot communication [10], [11], [12], [13]. However, it was reported that the introduction of encryption caused increment of overhead for data stream communication. There were significant increments of CPU utilization and the initialization of the $ROSTOPIC$ which shown peak usage, which was the same as during attack [14]. The variety of message types in ROS play important roles as huge data stream such as images and 3D laser data. The increment of packet size causing huge delay which affected overall performance. Measurement of performances were measured based on; i) overhead for the initial handshake, ii) pure transportation overhead and iii) overhead in a normal application [15]. Therefore, this paper aims to evaluate basic performance of CryptoROS architecture as proposed in [16] with a specific data type.

The rest of this paper is organised as follows: Section II addresses related works to data encryption for robot communications. Section III explains the overall structure of CryptoROS. The experimental setup, results and discussion are presented in Section IV. This paper is concluded in Section V.

II. Related Works

Secure communication channel could be achieved by enabling ROS-nodes to communicate with authenticity and confidentiality. Rodriguez et al. [11] figured out that an increment of overhead caused by the secure channel is only a small fraction of the load application itself generates. Further investigation was conducted on a robot's performance when ciphering the messages interchanged between ROS nodes under the publish/subscribe paradigm. Some other researchers showed that AES produced superior solution in implementation and required less CPU than other encryption algorithms [17], [18]. However, it is not recommended to imply a large data overhead in the network for multi-robot environments which required continuous interaction between each components. Morante et al. [12] discovered that, in humanoid robotics, a 1% overhead while respecting determinism in time can be acceptable if it means the devices can be less vulnerable to cyber attacks.

Some other encryption method were tested in securing robot communications. Amaran et al. [10] implemented a Constrained Application Protocol (CoAP) and MQ Telemetry

Transport for Sensor Nodes (MQTT-SN) which were designed for such devices. The outcome shown that MQTT-SN performs 30% faster than CoAP when transmitting the same payload [19], [20]. Rodr et al. [21] used a 3DES cyphering algorithm and performed system, evaluation in both computing and communications aspects. Experimental results showed that symmetric ciphers using private keys imposed significant delays [22], [23]. It is observed that a huge decreased of additional response time when using the secured solution for all tested robots. Mukhandi et al. [13] managed to secure robots' network communications by providing authentication and data encryption, therefore preventing man-in-the-middle and hijacking attacks.

III. CryptoROS Structure

The proposed architecture of CryptoROS has been described in Amini et al. [16]. It focused on providing peer-to-peer conversations between nodes, which capable of ensuring confidentiality and performing integrity violation check. A computation graph was set up for DoS attacks. The entities must also be scrutinized to ensure true identity. Nodes should not be allowed to publish/subscribe a topic or advertise a service without prior authorization. For evaluation of CryptoROS, three more components were added which consist of JSON Web Token, TLS and MySQL. Two machines were configured, running only the processes needed to carry out the experiments. A rudimentary attempt was also made to assess the overhead caused by CryptoROS.

A. JSON Web Token

The JSON Web Token (JWT) structure for CryptoROS is illustrated in Fig. 1. The header and payload in the form of UTF-8 byte array was encrypted using a Base64 encoding algorithm. The resulting strings were put together and fed into a hash function (SHA-256) to produce a message digest. This message was further encrypted using the private key (RSA) to create a signature. Then, the signature was encoded with a Base64 encoding algorithm and used to form the Access Token. Receiving entities that hold the associated public key (Managers) were able to reverse the procedure in order to ensure the Access Token has been issued by a trusted party (Authorization Server).

Nodes exchange a ROS connection header when establishing new connections. The ROS connection header holds crucial information regarding the connection that was established and used to route the connection. It was connected to ROS Publisher if the ROS connection header carries the topic field, or connected to ROS Service when ROS connection header carries the service field [24], [25]. The subscriber sends together with other data, its name (callerid), topic name to subscribe/connect (topic), and data type for the messages published to this topic (type). On a successful connection, the publisher replied with the fields shown in Fig. 2.

The Interceptor intercepts ROS connection header sent by its peer, extracted certain fields from it, and then compares with the values contained within the Access Token that had been sent by its peer. Then, it decided if the connection should be aborted or not. When issuing Access Tokens, users must decide how long the Access Token should last. Short-lived Access Tokens that expire after several hours or a couple of weeks are recommended. An Access Token cannot be revoked, so if it was issued with a short expiry time, the Manager is forced to refresh it frequently. However, users must select the best configuration which suit their needs.

B. TLS

To setup and run this project, OpenSSL was used to create private key and generate a Certificate Signing Request (CSR) that contains, among other data, a public key and a signature to prove ownership of the associated private key. The CSR was sent to a Certificate Authority (CA). The CA issued a certificate and gives all of the intermediate CA certificates needed to chain to the root CA certificate. The genrsa command was used to create RSA private key as shown below. The $-aes256$ option protected the private key with AES-256 and a passphrase. Henceforth, the encrypted private key can be securely backed up, transported, and stored. The last option, 2048, stated the size of the private key to create in bits. Fig. 3 and 4 show the implementation of these processes using genrsa and req commands respectively.

Eq. 1, 2, 3 and 4 show the process flow for key generation. In order to generate a CSR, the req command was used. Option $-new$ generated a new CSR by asking the user to provide the Distinguished Name (DN) field values. Value $-key$ provided the file name for OpenSSL tool to read the private key. As mentioned earlier, the CSR carried a public key and signed using the private key associated to the public key it hold. The $x509$ command was used to sign a CSR, resulting in the creation of certificate.

$$openssl\ genrsa - aes256 - out\ mng.key\ 2048 \tag{1}$$

$$openssl\ req - new - key\ mng.key\ - out\ mng.csr \tag{2}$$

$$\begin{gathered} openssl\ x509 - req - days\ 365 - in\ mng.csr \\ -textfile\ x509\ v3extensions.txt - CA\ root.crt \end{gathered} \tag{3}$$

$$-CAkey\ root.key\ -CAcreateserial\ -out\ mng.crt \tag{4}$$

The cipher suite configuration string used in this project is shown in Fig. 5. It selected the cipher suites that will be supported by the TLS server. TLSv1.2 keyword appends TLS 1.2 cipher suites to the list. As in Eq. 5, !LOW keyword permanently deletes all low strength encryption cipher suites from the list (e.g. 56-bit or 64-bit encryption algorithms). !MD5 keyword permanently deleted all cipher suites that used the obsolete and unsecure MD5 digest algorithm from the list. All cipher suites that do not checked for authentication with the involve parties (e.g. ADH and AECDH) were permanently deleted from the list using !aNULL keyword. The use of these cipher suites was strongly discouraged because they were vulnerable to a man-in-the-middle attack. !eNULL keyword permanently deleted all cipher suites that do not offer encryption from the list. !DES, !RC2, and !RC4 keywords

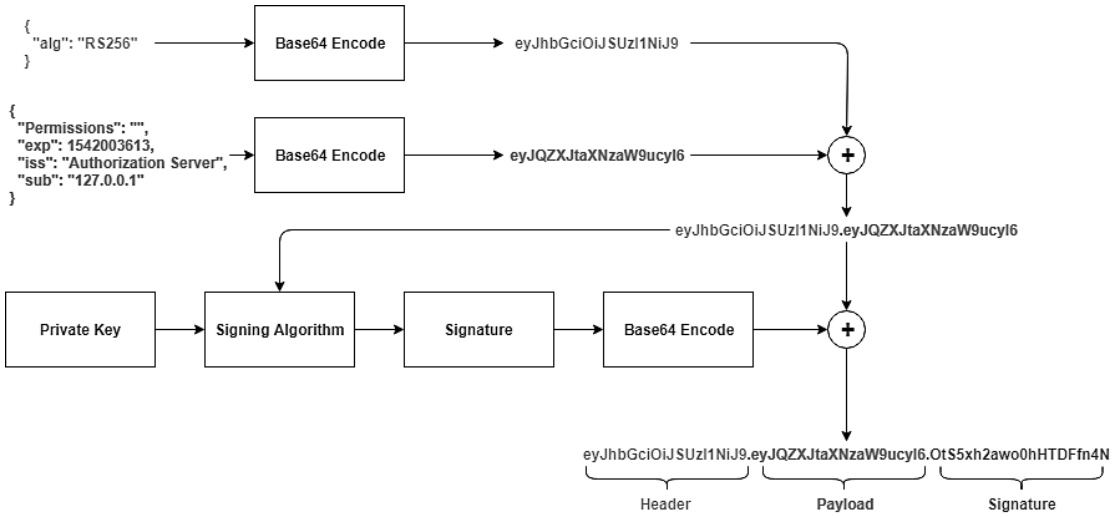

Fig. 1. JSON Web Token (JWT)

```
|.......callerid=/listener'...md5sum=992ce8a1687cec8c8bd883ec73ca41d1
...tcp_nodelay=0....topic=/chatter....type=std_msgs/String........callerid=/talker
...latching=0'...md5sum=992ce8a1687cec8c8bd883ec73ca41d1....message_definition=string data
....topic=/chatter....type=std_msgs/String
```

Fig. 2. ROS connection header. In red is the Subscriber, and in blue, the Publisher.

```
roham@roham:~$ openssl genrsa -aes256 -out egro1.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....................................................................+++
e is 65537 (0x10001)
Enter pass phrase for egro1.key:
Verifying - Enter pass phrase for egro1.key:
roham@roham:~$
```

Fig. 3. genrsa command

permanently deleted all cipher suites that used the obsolete and unsecure DES, RC2, and RC4 ciphers from the list, respectively. @STRENGTH keyword sorted the cipher suite list according to the cipher strength/key length [26].

$$\begin{aligned}"TLSv1.2 :! LOW :! EXPORT :! MD5 :! aNULL :\\ ! ADH :! AECDH :! DES :! eNULL : \\ ! RC2 :! RC4 : @STRENGTH" \end{aligned} \quad (5)$$

At the beginning the cipher suite list was empty, but the cipher suite list changed in some way as new keywords were added to the cipher suite configuration string. The cipher suite configuration string must be chosen carefully to avoid adding unsecure cipher suites to the list. The cipher command was invoked with the cipher suite configuration string as the parameter in order to list the cipher suites that match the requirements. As depicted in Fig. 6 the cipher suite names were very descriptive.

C. MySQL

Fig. 7 shows the Entity Relationship Diagram (ERD) for CryptoROS. The *user* table was used to store the Manager

```
roham@roham:~$ openssl req -new -key egro1.key -out egro1.csr
Enter pass phrase for egro1.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:MY
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:192.168.0.133
Email Address []:.

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
roham@roham:~$ openssl req -text -in egro1.csr -noout
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=MY, CN=192.168.0.133
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:a5:35:9f:a1:89:4c:e0:68:e2:9a:83:e8:c2:de:
                    62:2c:5d:3d:21:c9:98:b0:f7:2b:38:b2:a0:fb:87:
                    90:0b:be:c9:b3:68:46:79:1d:a3:a1:f5:a4:20:38:
                    6c:d5:96:bb:53:10:f3:7d:fb:f0:cf:35:0d:a8:81:
                    0f:3c:85:da:7c:7a:2e:e1:e8:62:39:84:14:30:44:
                    41:21:2f:ce:bb:b8:1e:65:13:35:a1:87:21:0b:2f:
                    09:09:32:b5:71:3a:3f:0b:20:e7:be:e3:c8:ff:4a:
                    cc:38:e1:8e:5d:ce:4d:2f:32:2c:39:b7:78:23:25:
                    34:73:54:3a:51:a9:55:b6:8e:12:f3:d9:dd:aa:12:
                    de:45:e1:01:d1:94:ef:59:9e:7d:af:c9:c9:91:c2:
                    98:8f:68:2a:0e:4a:a0:d3:2d:2d:cc:e3:49:cc:df:
                    a0:09:fe:a5:db:eb:d3:7f:28:ca:76:47:77:4e:ef:
                    a5:8c:b9:07:3f:9e:ed:62:fa:a3:57:79:10:10:80:
```

Fig. 4. req command

credentials. A Manager managed several publishers. This relationship was represented by a one-to-many relationship between the *user* table and the *publisher* table. The columns named *name*, *topic*, and *data_type* stored the name of the node, the (name of the) topic to publish messages to, and the data type of the messages published to this topic, respectively. The same applied to the subscribers, services, and service clients.

```
rohan@rohan:~$ openssl ciphers -v 'TLSv1.2:!LOW:!EXPORT:!MD5:!aNULL:!ADH:!AECDH:!DES:!eNULL:!RC2:!RC4:@STRENGTH'
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(256) Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(256)  Mac=SHA384
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(256)  Mac=SHA384
DH-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH/DSS   Au=DH   Enc=AESGCM(256) Mac=AEAD
DHE-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=DSS  Enc=AESGCM(256) Mac=AEAD
DH-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH/RSA   Au=DH   Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-SHA256    TLSv1.2 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA256
DHE-DSS-AES256-SHA256    TLSv1.2 Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA256
DH-RSA-AES256-SHA256     TLSv1.2 Kx=DH/RSA   Au=DH   Enc=AES(256)  Mac=SHA256
DH-DSS-AES256-SHA256     TLSv1.2 Kx=DH/DSS   Au=DH   Enc=AES(256)  Mac=SHA256
ECDH-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH/RSA Au=ECDH Enc=AESGCM(256) Mac=AEAD
ECDH-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AESGCM(256) Mac=AEAD
ECDH-RSA-AES256-SHA384   TLSv1.2 Kx=ECDH/RSA Au=ECDH Enc=AES(256)  Mac=SHA384
ECDH-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AES(256)  Mac=SHA384
AES256-GCM-SHA384        TLSv1.2 Kx=RSA      Au=RSA  Enc=AESGCM(256) Mac=AEAD
AES256-SHA256            TLSv1.2 Kx=RSA      Au=RSA  Enc=AES(256)  Mac=SHA256
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(128) Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(128) Mac=AEAD
ECDHE-RSA-AES128-SHA256  TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA256
ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(128)  Mac=SHA256
DH-DSS-AES128-GCM-SHA256 TLSv1.2 Kx=DH/DSS   Au=DH   Enc=AESGCM(128) Mac=AEAD
DHE-DSS-AES128-GCM-SHA256 TLSv1.2 Kx=DH       Au=DSS  Enc=AESGCM(128) Mac=AEAD
DH-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH/RSA   Au=DH   Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-SHA256    TLSv1.2 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA256
DHE-DSS-AES128-SHA256    TLSv1.2 Kx=DH       Au=DSS  Enc=AES(128)  Mac=SHA256
DH-RSA-AES128-SHA256     TLSv1.2 Kx=DH/RSA   Au=DH   Enc=AES(128)  Mac=SHA256
DH-DSS-AES128-SHA256     TLSv1.2 Kx=DH/DSS   Au=DH   Enc=AES(128)  Mac=SHA256
ECDH-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH/RSA Au=ECDH Enc=AESGCM(128) Mac=AEAD
ECDH-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AESGCM(128) Mac=AEAD
ECDH-RSA-AES128-SHA256   TLSv1.2 Kx=ECDH/RSA Au=ECDH Enc=AES(128)  Mac=SHA256
ECDH-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH/ECDSA Au=ECDH Enc=AES(128)  Mac=SHA256
AES128-GCM-SHA256        TLSv1.2 Kx=RSA      Au=RSA  Enc=AESGCM(128) Mac=AEAD
AES128-SHA256            TLSv1.2 Kx=RSA      Au=RSA  Enc=AES(128)  Mac=SHA256
rohan@rohan:~$
```

Fig. 5. Cipher suite configuration string

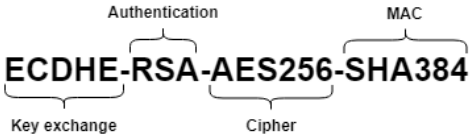

Fig. 6. Cipher suite name

IV. Results and Discussions

A. Experimental Setting

The purpose of this paper is to demonstrate how the proposed solution can alleviate the impact of various strategies used by malicious actors in order to exploit ROS based robots. The following experiments were carried out to analyze the performance impact of the Manager. All tests were executed on environment configurations consist of hardware, software, and network as listed in Table I.

In this experiment, a publisher named $talker$ was advertised its intention of publishing messages to a topic named $chatter$. A subscriber named $listener$ subscribed to topic $chatter$. Given the publisher's XML-RPC URI, the subscriber initiated, negotiated and established topic specific connection. The publisher began transmitting one hundred messages of type $std\_msgs/String$ afterwards, as shown in Fig. 8. The time command was used to launch the subscriber. When the program finished, time elapsed since its invocation was written to standard error by the time command [27]. The experiments were carried out 3 times with and without the proposed protocol intervention and the number of messages transmitted each time increased from 100, 250 and 500.

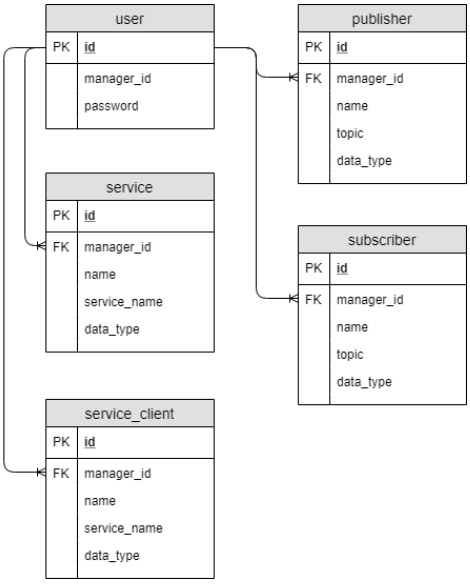

Fig. 7. Entity Relationship Diagram (ERD)

TABLE I. Evaluation Machine

CPU	Machine 1	Machine 2
Processor Number	Intel Core i5-6300HQ	Intel Core i5-3230M
Cores	4	2
Threads	4	4
Frequency	2.30 GHz - 3.20 GHz	2.60 GHz - 3.20 GHz
Memory (RAM)	8.00 GB	8.00 GB
Network	30 Mbps	30 Mbps

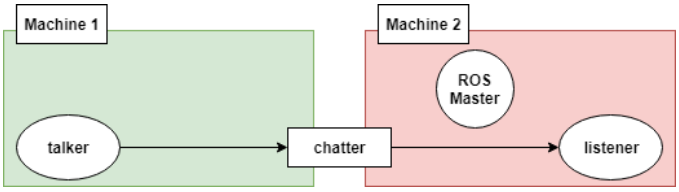

Fig. 8. Experimental setup

As mentioned previously, an attendee well versed in ROS was capable of controlling the ROS based robot without using the provided web based user interface during the DEF CON 20 conference [28]. These attacks required to a certain extent with little effort to accomplish. In the absence of the proposed protocol, data was transmitted in plain text format as depicted in Fig. 9. A TLS session was negotiated and established in accordance with the proposed protocol in order to secure the conversation as shown in Fig. 10.

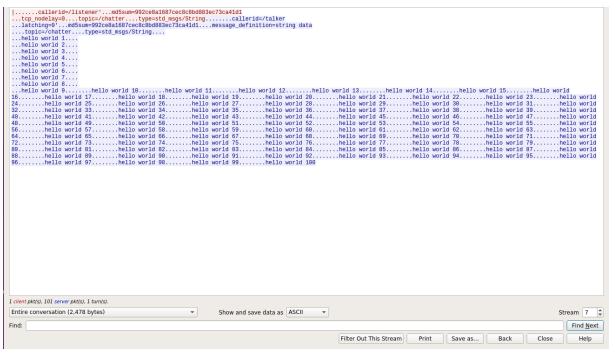

Fig. 9. Unencrypted data captured by Wireshark

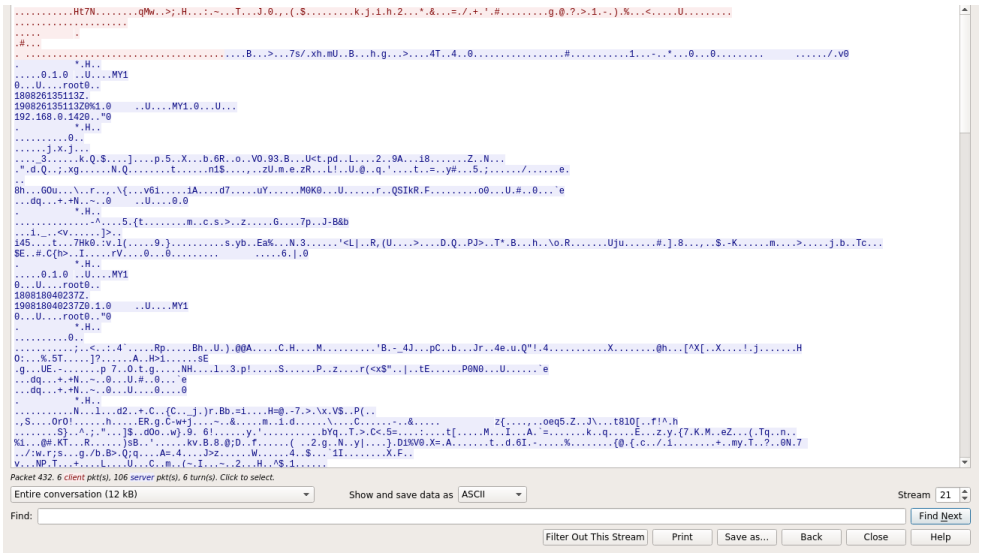

Fig. 10. Encrypted data captured by Wireshark

B. Performance Impact

The time elapsed values (Fig. 11, 12, 13) were obtained from launching the subscriber using the time command with and without the proposed protocol intervention. The summary of results in Table II concluded that the performance impact was inconsequential. To reduce the performance impact caused by the proposed protocol, the implementations were conducted using rich feature set offered by modern C++ (C++11 and C++14 standards). The finite computing resources used by class objects were also gracefully released (e.g. closing sockets) when appropriate to avoid running out of these resources. However, the size of the transferred data has increased from 2,478 bytes to 14,478 bytes. It happened based on the following factors; 1) many handshake messages required full handshake to negotiate and established a TLS session and 2) JWT was used for secure information transmission between the involved entities (e.g. publisher, subscriber).

The implementation of CryptoROS was capable in preventing unauthorized publishing and subscribing. The TLS handshake for inbound and outbound of peer-to-peer connection will failed and prohibit malicious nodes which were not supposed to be part of a specific conversation from injecting 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 nodes. This approach also made sure the messages, service requests and responses were not be disclosed to unauthorized persons (confidentiality). Any unauthorised intentional or accidental alteration of them was detected (integrity) and the proposed protocol ensured the authentication of involved entities.

TABLE II. AVERAGE ELAPSED TIME (S)

No. Messages	Without CryptoROS	With CryptoROS	Performance Impact
100	10.534	10.711	1.7%
250	25.588	25.713	0.5%
500	50.628	50.747	0.2%

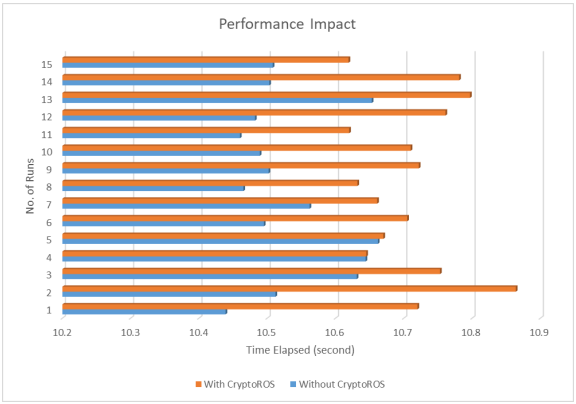

Fig. 11. Performance impact for 100 messages

V. CONCLUSION

CryptoROS has been designed in such a way that no changes needed to ROS software libraries and tools. Additionally, rebuilding nodes were not required to occupy secure conversation channels. CryptoROS works with all ROS client libraries regardless of the programming language they have been implemented. It can be concluded that CryptoROS does not increase much in computation time while providing security for peer robot communications. Further study should focused on evaluation in various types of data such as point cloud data and images.

ACKNOWLEDGMENT

The authors would like to thank Universiti Kebangsaan Malaysia for funding this research, grant Code: KRA-2018-007.

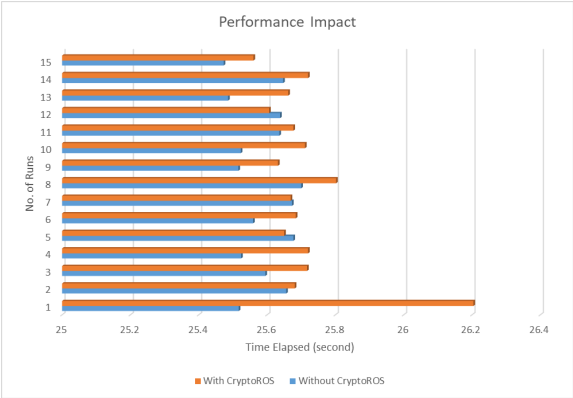

Fig. 12. Performance impact for 250 messages

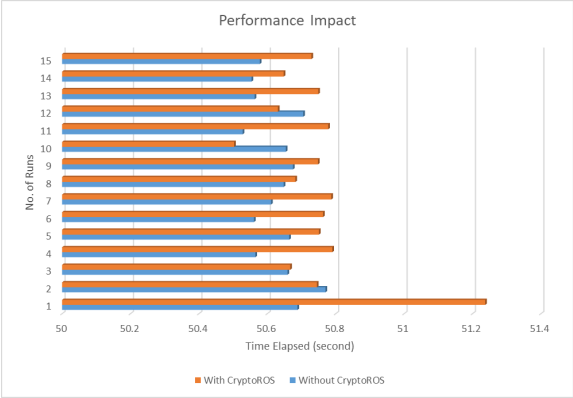

Fig. 13. Performance impact for 500 messages

REFERENCES

[1] N. F. A. Zainal, R. Din, N. A. Abd Majid, M. F. Nasrudin, and A. H. Abd Rahman, "Primary and Secondary School Students Perspective on Kolb-based STEM Module and Robotic Prototype," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 4-2, p. 1394, 2018.

[2] 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*, vol. 7, no. 6, pp. 3037–3045, 2017.

[3] I. Azmi, M. S. Shafei, M. F. Nasrudin, N. S. Sani, and A. H. A. Rahman, "Arucorsv: Robot localisation using artificial marker," in *Robot Intelligence Technology and Applications*, J.-H. Kim, H. Myung, and S.-M. Lee, Eds. Singapore: Springer Singapore, 2019, pp. 189–198.

[4] S. Pereira and M. S. Couceiro, "The Role of Security in Human-Robot Shared Environments : A Case Study in ROS-based Surveillance Robots," 2017.

[5] Bonaci T, "To Make a Robot Secure : An Experimental Analysis of Cyber Security Threats Against," pp. 1–11, 2015.

[6] F. Martín, E. Soriano, and J. M. Cañas, "Quantitative analysis of security in distributed robotic frameworks," *Robotics and Autonomous Systems*, vol. 100, pp. 95 – 107, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889017303044

[7] Y. Tang, D. Zhang, D. W. C. Ho, W. Yang, and B. Wang, "Event-based tracking control of mobile robot with denial-of-service attacks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–11, 2018.

[8] E. Basan, M. Medvedev, and S. Teterevyatnikov, "Analysis of the impact of denial of service attacks on the group of robots," in *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Oct 2018, pp. 63–638.

[9] 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, pp. 192–203, 2017. [Online]. Available: https://doi.org/10.1016/j.robot.2017.09.017

[10] M. H. Amaran, N. Arif, M. Noh, and M. S. Rohmad, "A Comparison of Lightweight Communication Protocols in Robotic Applications," *Procedia - Procedia Computer Science*, vol. 76, no. Iris, pp. 400–405, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.procs.2015.12.318

[11] F. J. Rodríguez-lera, "Message encryption in robot Operating system : collateral effects of hardening Mobile robots," vol. 5, no. March, pp. 1–12, 2018.

[12] S. Morante, J. G. Victores, and C. Balaguer, "Cryptobotics : why robots need cyber safety," vol. 2, no. September, pp. 23–26, 2015.

[13] M. Mukhandi, S. Pereira, and M. S. Couceiro, "A novel solution for securing robot communications based on the MQTT protocol and ROS," pp. 608–613, 2019.

[14] M. Horton and L. Chen, "Enhancing the Security of IoT Enabled Robotics : Protecting TurtleBot File System and Communication," 2017.

[15] B. B. . B. D. . P. Schartner, "Secure communication for the robot operating system," *IEEE International Systems Conference (SysCon)*, 2017.

[16] R. Amini, R. Sulaiman, and A. Hadi, "CryptoROS: A Secure Communication Architecture for ROS-Based Applications," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10, pp. 189–194, 2018.

[17] K. Tsai, Y. Huang, F. Leu, I. You, Y. Huang, and C. Tsai, "Aes-128 based secure low power communication for lorawan iot environments," *IEEE Access*, vol. 6, pp. 45 325–45 334, 2018.

[18] M. Horton, B. Samanta, C. Reid, L. Chen, and C. Kadlec, "Development of a secure, heterogeneous cloud robotics infrastructure: Implementing a mesh vpn and robotic file system security practices," in *SoutheastCon 2018*, April 2018, pp. 1–8.

[19] G. D. Salazar Ch, C. Venegas, and L. Marrone, "Mqtt-based prototype rover with vision-as-a-service (vaas)in an iot dual-stack scenario," in *2019 Sixth International Conference on eDemocracy eGovernment (ICEDEG)*, April 2019, pp. 344–349.

[20] P. Singhal, P. Sharma, and B. Hazela, "End-to-end message authentication using coap over iot," in *International Conference on Innovative Computing and Communications*, S. Bhattacharyya, A. E. Hassanien, D. Gupta, A. Khanna, and I. Pan, Eds. Singapore: Springer Singapore, 2019, pp. 279–288.

[21] F. J. Rodr, F. Casado, C. Fern, and F. Mart, "Cybersecurity in Autonomous Systems : Evaluating the performance of hardening ROS," no. June, 2016.

[22] M. Nazeh, A. Wahid, A. Ali, B. Esparham, and M. Marwan, "A Comparison of Cryptographic Algorithms : DES , 3DES , AES , RSA and Blowfish for Guessing Attacks Prevention," *Journal of Computer Science Applications and Information Technology*, vol. 3, no. 2, pp. 1–7, 2018.

[23] V. DiLuoffo, W. R. Michalson, and B. Sunar, "Robot Operating System 2: The need for a holistic security approach to robotic architectures," *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, pp. 1–15, 2018.

[24] B. Alexander, "[online] ROS Connection Header," 2011.

[25] D. Thomas, "[online] wiki: Parameter Server," 2013.

[26] O. S. Foundation, "[online] openssl.org Cipher," 2015.

[27] M. Kerrisk, "[online] Time man7.org." 2018.

[28] J. McClean, C. Stull, C. Farrar, and D. Mascareñas, "A preliminary cyber-physical security assessment of the robot operating system (ros)," 2013. [Online]. Available: https://doi.org/10.1117/12.2016189