Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Energy-aware dynamic Internet of Things security system based on Elliptic Curve Cryptography and Message Queue Telemetry Transport protocol for mitigating Replay attacks

Floriano De Rango [*], Giuseppe Potrino, Mauro Tropea, Peppino Fazio

University of Calabria, DIMES Department, P. Bucci 39/c, 87036 Rende (CS), Italy

A R T I C L E   I N F O

Article history:
Received 30 November 2018
Received in revised form 12 October 2019
Accepted 14 October 2019
Available online 25 November 2019

Keywords:
IoT
ECC
MQTT
Replay attacks
IDS

A B S T R A C T

Security in the context of Internet of Things (IoT) is a critical challenge. The purpose of this work is to model and evaluate a dynamic IoT security system based on a generic IoT edge network in which nodes exchange messages through the Message Queue Telemetry Transport (MQTT) protocol. This work aims to increase MQTT security by mitigating data tampering, eavesdropping and Replay attacks by using the Elliptic Curve Cryptography (ECC), timestamps and wake up patterns, with the purpose of preserving node energy. The evaluated results will show that it is possible to increase the system lifetime by linking security levels and energy.

1. Introduction

The term IoT was born as a title of a presentation made by Kevin Ashton in 1999 at MIT [1]. IoT overlaps between Mobile Computing, Pervasive Computing, Wireless Sensor Networks and Cyber–Physical System. IoT can be defined as a wired or wireless network constituted by connected and unequivocally identified devices able to process data and communicate with each other with or without human help. Nowadays, the IoT is used in many fields for example: Logistic, Smart environment, Smart Agriculture, Smart cities etc. By the end of 2020 it is estimated that there will be approximately 30 billion connected devices with a data exchange greater than 40 Zettabytes [2]. These expectations raise many IoT challenges such as scalability, data volumes, self-organizing, data interpretation, interoperability, automatic discovery, software complexity, security and privacy, wireless communication. Since sensors typically have limited computing and storage capabilities, it is common to forward the generated data to a cloud computing platform for data processing and analysis, but the network latency and jitter can become significant. As a result, the low latency requirement of various delay-sensitive applications such as vehicular networks and smart health services cannot be met. To address the above issue, the Mobile Edge Computing (MEC) paradigm has emerged in recent years. MEC is an essential paradigm shift toward a hierarchical architecture and a more responsive design. As shown in Fig. 1, the edge is an intermediate computing layer between the cloud and end devices that complements the advantages of cloud computing by performing data processing at the edge of radio access networks of mobile telecommunications networks [3,4]. Although the fog computing paradigm offers many benefits for different IoT applications, it faces many challenges such as scalability, complexity, dynamicity, heterogeneity, latency and security [5].

[*] Corresponding author.
E-mail addresses: derango@dimes.unical.it (F. De Rango), giuseppe.potrino@unical.it (G. Potrino), m.tropea@dimes.unical.it (M. Tropea), p.fazio@dimes.unical.it (P. Fazio).

https://doi.org/10.1016/j.pmcj.2019.101105
1574-1192/© 2019 Elsevier B.V. All rights reserved.

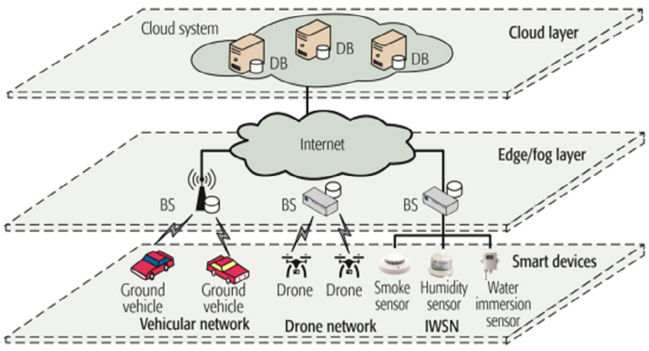


Fig. 1. Three tier architecture [3].

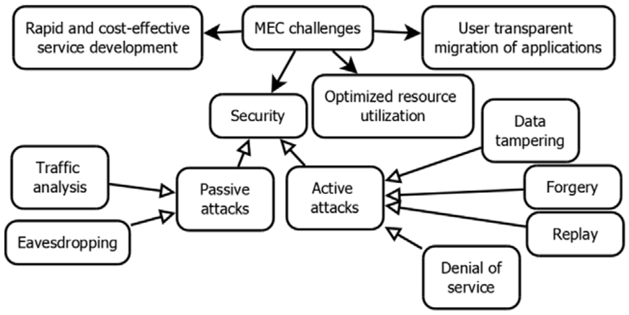


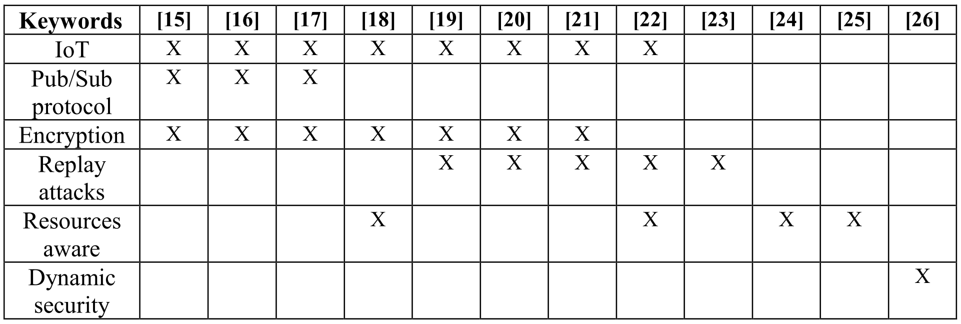
Fig. 2. Edge computing main vulnerabilities.

One of the critical requirements of supporting IoT applications in MEC is security, because in wireless communications the medium can be accessed by both authorized users and adversaries; consequently, this makes it vulnerable to many security attacks [6]. Edge computing and fog computing terms are used interchangeably in both academia and industry. Their main objectives are the same, but they differ on how they process and handle the data and where the intelligence and computing power are placed [7]. This work aims to increase the security of an IoT context at edge layer in which there are a certain number of lightweight nodes exchanging messages by using the MQTT protocol through a broker which is placed on a fog node. It is proposed to mitigate data tampering and eavesdropping by applying the ECC on the MQTT payloads. As a higher key-strength causes more energy consumption than a lower one, the used security level will be dynamically changed, and it will be linked to the node energy level in an attempt to increase the system lifetime. Obviously, a low security level key must be changed with high frequency for guaranteeing good security. A mechanism is also proposed to mitigate Replay attacks by adding a timestamp to the ciphered payloads and by using some wake up patterns to preserve lightweight nodes energy by decreasing the received replicated messages that must be dropped. A Replay attack can be detected by an IDS that is placed on each network node which controls the packet timestamp.

2. Preliminaries

2.1. Edge computing and its vulnerabilities

The successful deployment of MEC still has to face a number of open research challenges due to undefined incentives for the network service provider and the intrinsic limitations of wireless networks [3]. In fact, the MEC users remain in the coverage area of the MEC service provider for a limited time and they ask for variety of services that change rapidly, the resources available on the MEC server are limited compared to the Cloud server [8–10]. However in [11] authors emphasize the benefits that can be obtained through the distributed analytics at edge in terms of energy consumption and delay reduction through the fog gateway. Moreover, in [12] authors show the potential risks present in a cloud system where off-loading policy has been applied. One possible risk is represented by timing attacks that could be faced by considering random delays in the cryptographic algorithm mitigating the information leakage. In the edge-based IoT system various security threats exist in which an adversary can make both passive and active attacks. In fact, an adversary can passively monitor network communication for analytics, tamper with the transmitting or stored data, construct new packets, replay intercepted packets or violate the availability of resources [6] (see Fig. 2).



Keywords	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]
IoT	X	X	X	X	X	X	X	X				
Pub/Sub protocol	X	X	X									
Encryption	X	X	X	X	X	X	X					
Replay attacks					X	X	X	X	X			
Resources aware				X				X		X	X	
Dynamic security												X

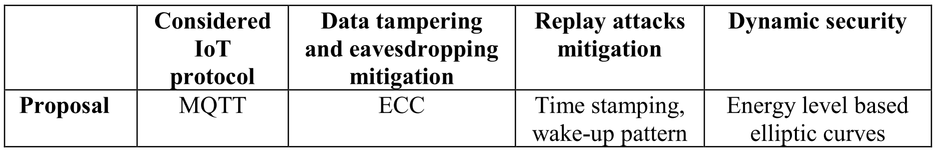
Fig. 3. Related works summary.

2.2. Related works on security risk mitigation in IoT

In [13] authors analyzed the weakness of the authentication scheme proposed in [14] where IoT devices with limited resources have been considered. Moreover, authors show how applying the ECC and an enhanced authentication scheme they can provide robustness against Replay attacks and Stolen session key attacks. In [15] a new approach to communication and security key management, which is based on the MQTT protocol, was proposed. In the encryption/decryption phase, the authors used two different approaches (RSA and Elliptic Curve) in order to give the advantage and the benefit of each of them. In [16] a novel lightweight security solution for publish–subscribe based protocol in an IoT Fog network using ECC, a scalable key exchange mechanism and an efficient and responsive mutual authentication, and encryption system, was proposed. In [17] the Secure-MQTT (SMQTT) protocol was proposed with the purpose of increasing the security of the MQTT protocol. In particular, the Publish messages are replaced by a new message type called SPublish which is identified with the reserved MQTT header code "0000". In this case, the publisher sends a SPublish message containing a payload which is ciphered by using the destination ECC public key. In [18] a privacy-preserving smart parking system based on IoT elliptic curve as ECC is an ideal candidate for implementation on devices where the major computational resources are limited, was proposed. Instead, in [19] a system that can withstand replay attacks, man-in-the-middle attacks and wiretapped secret key attacks was proposed. Attackers can intercept packets that are transmitted during the authentication and distribution of the session key between sensor nodes, and then replay them after some time. As the proposed system generates a new nonce value for every authentication attempt, no authentication is established if an attacker replays the packets after a time interval. [20] proposes a Key Management Protocol for mobile and industrial IoT systems, targeting, at the same time, robust key negotiation, lightweight node authentication, fast re-keying, and efficient protection against replay attacks. In [21] a lightweight IoT security protocol based on an asynchronous One Time Password (OTP) was proposed. The OTP is a password that can be used only one time. It is based on pre-shared key and a random challenge in the case of asynchronous OTP. The random challenge protects the authentication against replay attacks and cryptanalysis attacks. In [22] a lightweight and reliable framework was proposed which uses a combination of universally unique identifier, timestamp and a self-learning battery depletion monitor to detect and mitigate replay attacks on battery dependent IoT devices. In [23] a new security model to address cloning attack, MITM attack and Replay attack was proposed. The concept of zero knowledge protocol was used which ensures non-transmission of crucial information between the prover and the verifier. In [24,25] a trust management system to detect the anomalous and malicious behavior of nodes in a MANET network was proposed and an energetic analysis is also performed to understand the energy consumption due to additional introduced security mechanisms. In [26] a dynamic security solution to avoid denial of service attack without increasing the complexity of wireless mobile devices was proposed. Fig. 3 summarizes the described related works.

2.3. Work objectives

This work proposes to mitigate data tampering, eavesdropping and replay attacks in an IoT context. Data tampering and eavesdropping are generally mitigated by using cryptography. Nowadays a very used security system is SSL/TLS, but its use in an IoT context can consume a lot of device energy. In [15–18] the use of ECC in an IoT context was proposed because it is an ideal candidate for implementation on lightweight devices. The proposed system, similarly to [15–17], wants to apply ECC to an IoT fog network. Specifically, this work wants to mitigate these vulnerabilities on a MQTT based communication. We use a mechanism similar to the proposal of [17] for recognizing ciphered packets by non-ciphered ones. Unlike the previous referred works, we propose to change the used elliptic curve (and the relative key-strength) dynamically (the dynamic security concept was also used in [26]) on the basis of the residual node energy to increase the system lifetime. In fact, also if ECC is lightweight, it also needs a significant amount of computation and energy (like all cryptography techniques), and higher is the key strength and higher is the consumed power. In fact, as specified in [24,25] the use of a security system needs of additional energy consumption. To maintain a high security level at elliptic



	Considered IoT protocol	Data tampering and eavesdropping mitigation	Replay attacks mitigation	Dynamic security
Proposal	MQTT	ECC	Time stamping, wake-up pattern	Energy level based elliptic curves

Fig. 4. Proposal summary.

curve key-strength reduction, we propose to tune the key change frequency. In [19,23] some Replay attacks mitigations were proposed; however, the lightweight nodes energy problem was not considered. In [20,21] some lightweight IoT security protocol were proposed but the lightweight nodes energy waste caused by an attacker was not studied. In [22] a lightweight Replay attack detection framework for battery dependent IoT devices was proposed, in which the battery level was monitored for Replay attacks detection. Unlike the works in [19–21,23], the proposed system wants to mitigate the Replay attacks by limiting the energy waste that can be caused by an attacker. We also propose to reduce the energy waste in advance, while the work in [22] discovers the attack on the basis of the battery level trend. There are many techniques for facing Replay attacks, for example the use of one-time password, nonces and MAC or timestamp. The advantages of the use of timestamps is that it is not needed to generate random numbers which have to be exchanged between nodes. In networks that are unidirectional or near unidirectional, this can be an advantage because they do not need to first exchange nonces etc. The considered context is near unidirectional because we consider sensor nodes that send message to the broker which forwards them to actuators, then we can advantage of the timestamp usage. The proposed system wants to mitigate Replay attacks by using a timestamp which must be appended to the MQTT packet payload. In addition to using the timestamp, in order to avoid energy waste on lightweight nodes in Replay attacks, it is proposed to use some wake-up patterns to reduce the number of received replicated messages. In fact, the dropping process causes an extra energy use that can be important for battery based lightweight nodes. These patterns synchronize each lightweight node with the broker. Fig. 4 summarizes the proposal of this work.

3. Key elements in the considered application domain

3.1. Reference context

The system which is proposed in this work is collocated in an IoT context. In particular, it is collocated between the Smart devices Layer and the Fog/Edge Computing Layer. Then, the system is constituted by a certain number of lightweight nodes, each with a sensor or actuator module, and a central fog node. Monitored data by sensor nodes are sent periodically to the fog node by using the MQTT protocol. These messages are sent by the fog node to the actuator nodes. The system has a star topology. The broker is situated on the fog node, which has a power source and more processing and storing capacity. Each node has a Host IDS for detecting replicated messages which are sent periodically by an attacker. Then, the main context parameters are related to node number, percentage of sensor/actuator nodes, messages exchange frequency, node battery and frequency of replicated packets.

3.2. A lightway protocol in IoT: MQTT protocol

In the IoT technology, there are various used protocols like MQTT, MQTT-SN and CoAP. In [27] it was observed that when the packet loss rate is low, MQTT delivers messages with lower delay than CoAP. We consider the use of the MQTT protocol in this work. MQTT [28] is a lightweight application level protocol generally used in IoT systems. Network nodes communicate through an MQTT broker. This protocol uses an asynchronous communication based on publish/subscribe model in which data flow is decomposed in topics. The protocol messages have lightweight header overhead and they are generally used with TCP, 6LoWPAN or UDP. The protocol architecture consists of several clients that can be either the publisher or subscriber of some topics (queue of messages) contained on a broker node which acts as a server. Each client, before starting a communication, needs to start a session with the broker and then it can make a subscription to interesting topics or publish messages.

The main message types that are exchanged in this protocol are (Fig. 5): CONNECT (it establishes a connection between a client and the broker), PUBLISH (it publishes data on a specified topic), SUBSCRIBE (it creates a subscription to a specified topic), UNSUBSCRIBE (it cancels a subscription from a specified topic), DISCONNECTED (it closes an established connection with the broker), CONNACK, PUBACK, SUBACK, UNSUBACK. In our context, the sensor nodes send their sensed data in Publish messages to some MQTT topics to which the actuator nodes are subscribed. Then, we also need to define the MQTT context topics.

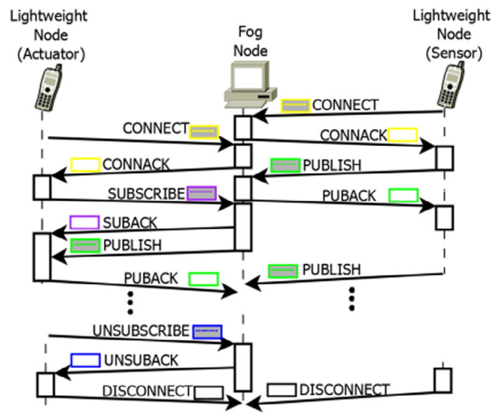


Fig. 5. MQTT protocol messages.

Key parameters
Number of lightweight nodes
Percentage of sensor and actuator nodes
Device battery capacity
Interval between two lightweight node messages sending
Interval between two replicated packets by attacker
MQTT topics
Used elliptic curves for various energy levels
Key exchange intervals related to the various security levels

Fig. 6. MQTT protocol messages.

3.3. A lightweight cryptography in IoT: Elliptic curve cryptography

To face data tampering and eavesdropping challenges in a protocol such as MQTT, it is necessary to use cryptography. It regards the study of techniques for secure communication in the presence of adversaries [29] and it offers authentication, confidentiality, availability, integrity and access control [30]. It is possible to increase the security of the MQTT protocol by using Secure Socket Layer/Transport Layer Security (SSL/TLS) [31] but it can consume a significant amount of energy in an IoT device. Elliptic curve cryptosystems over finite field have some benefits like the key size that can be considerably smaller compared to additional cryptosystems like RSA, Diffie–Hellman since only exponential time attack is known so far if the curve is carefully chosen [32–36] and ECC depends on the difficulty of explaining the *Elliptic Curve Discrete Logarithm Problem* (ECDLP). Standards for Efficient Cryptography (SEC) [37] define F_p and $F_2{}^m$ elliptic curve types. The elliptic curves on $F_2{}^m$ are considered in this work. The domain parameters of a curve on $F_2{}^m$ are a sextuple T composed as in the follow:

$$T = (m, f(x), a, b, G, n, h)$$

where m is an integer number specifying a finite field $F_2{}^m$, *f(x)* is an irreducible binary polynomial of grade m, specifying the polynomial base representing $F_2{}^m$, a and b belong to $F_2{}^m$ and they specify an elliptic curve $E(F_2{}^m)$ defined by the equation E: $y^2 + xy = x^3 + ax^2 + b$, $G = (x_G, y_G)$ is a base point on $E(F_2{}^m)$, n is a prime indicating the order of G and h is an integer that is the cofactor $h = \#E(F_2{}^m)/n$. In our work, we need to choose the elliptic curves and the relative key exchange frequencies to use for the various energy levels.

3.4. Key parameters summarization

Fig. 6 summarizes the aforementioned key context parameters.

Level	Curve	Field	f(x)
High	Sect409r1	$F_2{}^{409}$	$x^{409} + x^{87} + 1$
Medium	Sect239k1	$F_2{}^{239}$	$x^{239} + x^{36} + 1$
Low	Sect193r2	$F_2{}^{193}$	$x^{193} + x^{15} + 1$

Fig. 7. Used elliptic curves.

Name	Description
Normal topic	It is a generic MQTT topic (nodes send pub/sub messages in clear)
Cipher topic	It is a topic that admits only ciphered messages. ECC is applied to payloads. The SPUBLISH messages are identified by using the reserved MQTT header code "0000" while the SPUBACK messages are identified by using the reserved MQTT header code "1111". This mechanism protects against eavesdropping and data tampering.
Authenticated Cipher topic	It is like the Cipher topic but it provides an authentication mechanism. Accepted credentials are stored in the broker. After the authentication of each node, a token is generated and is exchanged in each message by using cryptography.

Fig. 8. Used topic.

4. Proposed dynamic IoT security system over EDGE/FOG node

4.1. Security levels of elliptic curves

This section explains the key parameters relating to the used elliptic curves and their key exchange intervals for the considered security levels. This work considers three different security levels summarized in Fig. 7. All the domain parameters details of these curves can be found in [37].

An elliptic curve with a key length of $2 * t$ bits has a key strength of approximately t bits. Then, approximately 2^t operations are necessary to break the discrete logarithm problem associated to the relative elliptic curve. This means that, in order to break an elliptic curve key, it is necessary a time which depends on the needed number of operations and from the current computing power. If we can establish that an interval T_0 provides a good security for the LOW security level (the key of the elliptic curve of the low security level cannot be broken in a T_0 period), then, the interval for the MEDIUM and the HIGH security level needs to be computed. A key strength of $t + 1$ bit needs almost double (we consider $b = 1{,}5$) number of operations to be broken respect than a key strength of t bit. Then, the interval for having secure MEDIUM and HIGH security level can be computed as in the following:

$$T_i = T_0 * b\^{}k$$

where T_i is the interval for the current security level (MEDIUM or HIGH), k is the extra key strength (in bit) respect to the LOW security level (for example if we consider MEDIUM security level we have k = (239 − 193)/2). This means that lower is the security level of the key and higher must be its change frequency. As seen, the low security level uses a key length of 193 bits which has a key strength of 96 bits that it is acceptable until 2020. After 2020 this key strength may become insufficient and, for this reason, in this work this key is changed every 5 min (T_0). The other key exchange intervals are computed starting from $T_0 = 5$ min as specified, obtaining 39 days for medium security level and humanly never for high one. The medium security level is acceptable until 2030, while the high one is also acceptable for the following decades [38]. The keys exchange can be susceptible to Man-in-the-Middle attack. We need a Certification Authority (CA) to mitigate this attack and each node must know the CA public key. In this scenario, the high security key can be static and then can be certified by a CA and used in the exchange of medium and low security keys.

4.2. MQTT topics and dynamic keys exchange

This section describes the considered MQTT topics and the used key exchange protocol. The broker in our context contains three different topic types distinguished by name prefix. Each topic type has specific needs and security levels, which can be summarized in Fig. 8. It is assumed that all nodes are synchronized.

The broker and each lightweight node that uses a ciphered topic have a private key and a public key which are certified by a Certification Authority. The certificated Public Key of each node is the high security key. At the system beginning each node using ciphered topics uses this certified key. Subsequently, when the lightweight node energy becomes slow, it starts to use some lightweight temporary security keys which must be changed frequently on the basis of its current security level. These temporary keys (medium and low security) are exchanged by using the certified key for avoiding

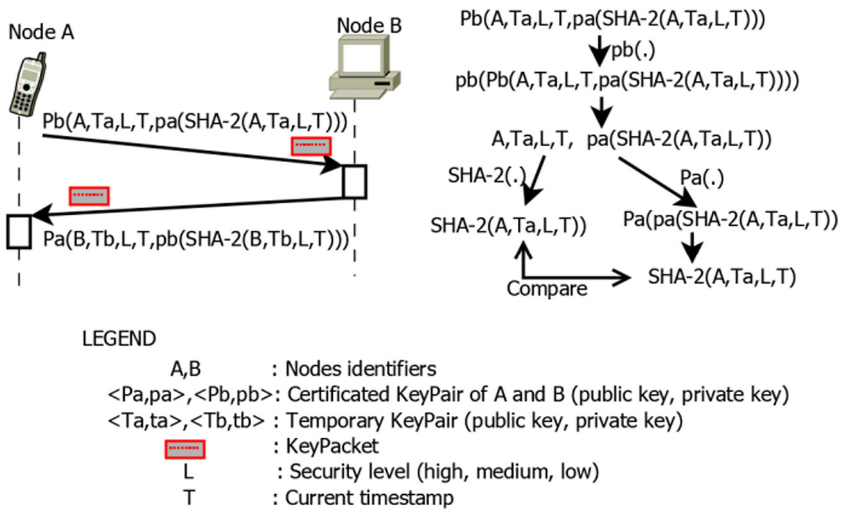

Node A
Node B
Pb(A,Ta,L,T,pa(SHA-2(A,Ta,L,T))
Pa(B,Tb,L,T,pb(SHA-2(B,Tb,L,T))
Pb(A,Ta,L,T,pa(SHA-2(A,Ta,L,T))
pb(.)
pb(Pb(A,Ta,L,T,pa(SHA-2(A,Ta,L,T))))
A,Ta,L,T, pa(SHA-2(A,Ta,L,T))
SHA-2(.)
Pa(.)
SHA-2(A,Ta,L,T))
Pa(pa(SHA-2(A,Ta,L,T))
SHA-2(A,Ta,L,T)
Compare
LEGEND
A,B : Nodes identifiers
<Pa,pa>,<Pb,pb>: Certificated KeyPair of A and B (public key, private key)
<Ta,ta>,<Tb,tb> : Temporary KeyPair (public key, private key)
: KeyPacket
L : Security level (high, medium, low)
T : Current timestamp

Fig. 9. Key exchange protocol.

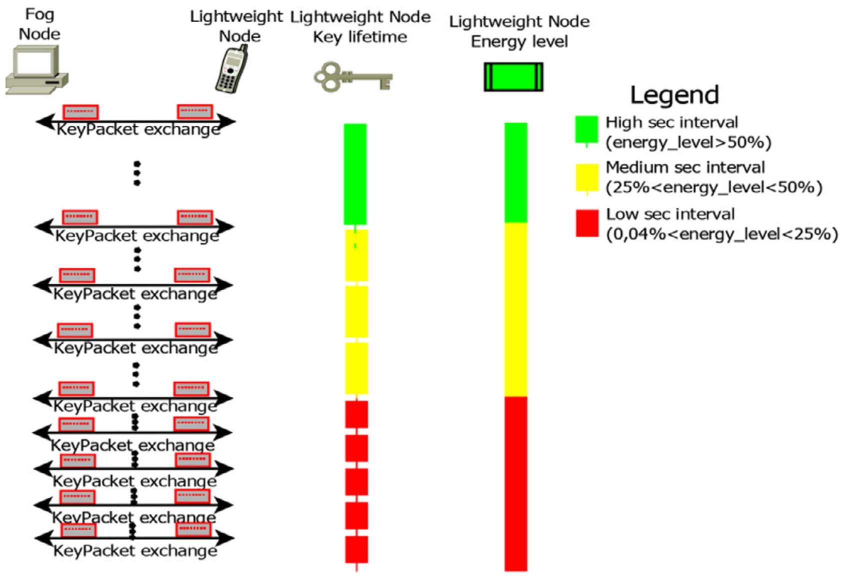
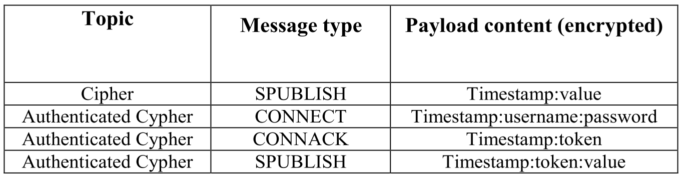

Fog Node
Lightweight Node
Lightweight Node Key lifetime
Lightweight Node Energy level
KeyPacket exchange
Legend
High sec interval (energy_level>50%)
Medium sec interval (25%<energy_level<50%)
Low sec interval (0,04%<energy_level<25%)

Fig. 10. Dynamic key exchange frequency.

some Man-in-the-middle attacks in which an attacker can replace a node's public key with its own one. The key exchange protocol is described in Fig. 9:

Fig. 9 describes how the temporary public keys are exchanged in a secure manner. In fact, the application of destination public key guarantees confidentiality, the use of a hash function guarantees integrity, the application of the source private key guarantees authentication of the temporary public key just sent and the timestamp is useful to avoid Replay attacks. The fog node has three key pairs (of which the certified one) corresponding to three security levels which can be used by different nodes on the basis of their energy level. Before starting to communicate, when the lightweight node energy has become low, it has to send its current temporary public key to the fog node. Each ciphered communication starts with a KeyPacket exchange. This packet contains the used security level which can be LOW, MEDIUM or HIGH and the Public Key of the source node (Fig. 9). The fog node responds to this packet by sending one of its current three Public Keys (that are relative to the received security level). At each key change, the source node informs the destinations by sending the new Public Key and its relative level. Each lightweight node uses, at the beginning, a HIGH security level. When the lightweight node energy level becomes lower than 50%, the security level switches from high to MEDIUM; however, in order to have a good security level, it is necessary to increase the keys exchange frequency (as previously specified). Similarly, when the lightweight node energy level becomes lower than 25%, the security level switches from MEDIUM to



Topic	Message type	Payload content (encrypted)
Cipher	SPUBLISH	Timestamp:value
Authenticated Cypher	CONNECT	Timestamp:username:password
Authenticated Cypher	CONNACK	Timestamp:token
Authenticated Cypher	SPUBLISH	Timestamp:token:value

Fig. 11. Main payload contents.

LOW with a consequent increasing of the keys exchange frequency. Then, lower is the security level and higher is the number of exchanged Key Packet, as shown in Fig. 10.

This mechanism, which is reassumed in the following pseudo-code, can guarantee a system lifetime increase. The "security_level" variable represents the node current security level and can be HIGH, MEDIUM or LOW. The "last_change" variable represents the time of the last key change. The "energy_level" variable represents the node current energy level in milli-Ampere. The "k" variable represents the current generated public key that must be sent to the fog node. The called functions and the used constants are self-explanatory. To evaluate the performance of this proposal, we need to measure the energy consumption of the lightweight nodes considering the various hardware modules and the generated overhead related to the frequency of the key change.

```
security_level=HIGH_SEC;
while(energy_level>0,5*BATTERY_CAPACITY){
    //maintain the current security level
}
last_change= CURRENT_TIME;
security_level=MEDIUM_SEC;
while(energy_level>0,25*BATTERY_CAPACITY){
    if(CURRENT_TIME – last_change> MEDIUM_SEC_INTERVAL){
        k=generate_new_key(security_level);
        send_to_fog_node(k);
        last_change= CURRENT_TIME;}}
security_level=LOW_SEC;
while(energy_level>0,04*BATTERY_CAPACITY){
    if(CURRENT_TIME – last_change> LOW_SEC_INTERVAL){
        k=generate_new_key(security_level);
        send_to_fog_node(k);
        last_change= CURRENT_TIME;}}
turn_the_node_off();
```

4.3. Replay attacks to the fog node mitigation

Once the operation to update the keys has been established, it is necessary to explain how the main ciphered packets are constructed (see Fig. 11).

In an Authenticated Cipher topic, at the beginning a lightweight node sends a CONNECT packet containing username and password and if it is authenticated it will receive a token which can be used to authenticate SPUBLISH messages.

Each node IDS controls the correctness of the received timestamp to detect a Reply attack. All these previous described packet formats are exchanged in a ciphered manner. This means that an attacker cannot read the payload content because it does not know the destination private key. Moreover, if the attacker tries to modify a ciphered payload, this will reach

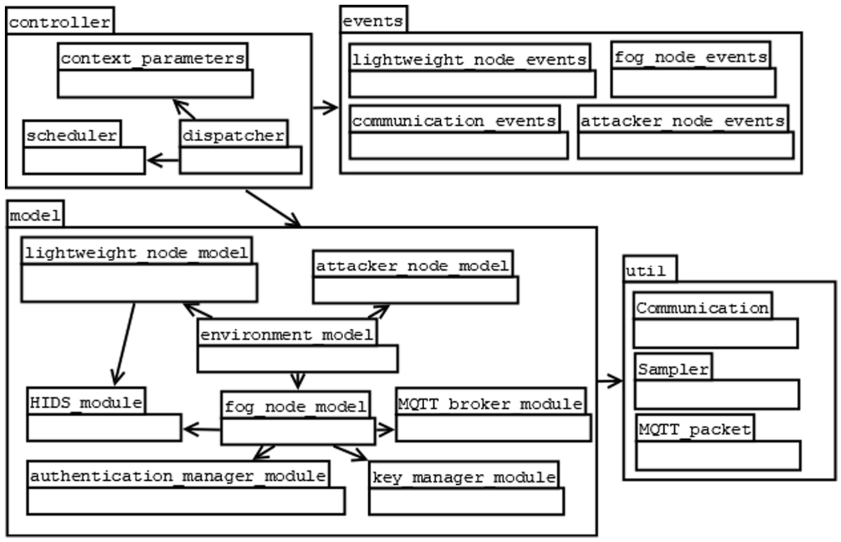


Fig. 12. UML of implemented simulator.

the destination in a corrupted manner because its decryption will return a strange format and then will be dropped. Furthermore, if an attacker tries to forge a packet with a timestamp of an authenticated topic, this will also be dropped because the attacker does not know a valid token if it has not been authenticated (obviously the attacker does not know the credentials to obtain a token, which also is exchanged in a ciphered manner in other nodes authentications). Then, given a source, all its sent packets will contain a different and increasing timestamp that represents the current time at which the packet was forged. If an attacker replays a packet, the destination will notice that the received packet does not contain a timestamp greater than the last received one by the same source and then it will drop the packet. For each connected lightweight node, the fog node stores the respective timestamp contained into the last received packet. Then, if the fog node receives a packet having a timestamp older or equals than those stored one and coming from the same node, it drops the received packet. If the fog node receives a packet having a newer timestamp than the stored one and coming from the same node, it accepts the packet and updates the stored timestamp with the received last one. This mechanism can be reassumed in the following pseudocode. The "map" data structure contains for each node the timestamp of the last received packet. The "pck" variable represents the current elaborating packet and contains its source, the timestamp and the value. The called functions are self-explanatory.

```
Map<Node,Timestamp> map;
initialize(map);
loop(){
    pck=receive_packet();
    if(map.contains(pck.source_node) &&
    map.get(pck.source_node)>=pck.timestamp){
        drop_packet(pck);
    }else{
        elaborate_packet(pck);
            map.put(pck.source_node,pck.timestamp);}
  }
```

4.4. Replay attacks to the lightweight nodes mitigation

Each lightweight node stores the last received timestamp by the broker node which is used similarly to those stored in the broker. In order to reduce the number of received replicated packets (that can waste energy), each lightweight node agrees a wake-up frequency with the broker. At each wake-up, each lightweight node remains active for a fixed time (we call it *active_interval*) for receiving packets. Then, each lightweight sensor node wakes-up at each established time to send the last measured value to the broker. The broker stores the last updated value for each topic and sends it to the interested actuator nodes at their wake-up times. Each lightweight actuator node wakes up for *active_interval* at each established time. The following pseudo-code reassumes a generic lightweight node wake-up pattern. The broker knows the wake-up frequency of each lightweight node, because it depends on its security level (the broker knows the security level of each node because it is established in the last received KeyPacket), and the timestamps of each exchanged packet. In particular, higher is the security level and higher is the wake-up frequency because this means that the lightweight node has a high energy level. We consider three fixed wake-up frequencies, each for each security level.

```
  timestamp=0;
  wake_up_period=compute_wake_up_period(CURRENT_SECURITY_LEVEL);
  loop(){
      wake_up_time=CURRENT_TIME;
      while(CURRENT_TIME – wake_up_time< ACTIVE_INTERVAL){
       pck=receive_packet();
       if(timestamp>=pck.timestamp){
                 drop_packet(pck);
       }else{    elaborate_packet(pck);
                 timestamp=pck.timestamp;}
  }
  sleep(wake_up_period);
}
```

The “timestamp” variable represents the timestamp of the last received packet from the fog node. The “wake_up_ period” variable represents the interval that must elapse between two node activations. The “wake_up_time” variable represents the time of the last activation. The “pck” variable is similar to the previous pseudo-code. The called functions are self-explanatory

5. Performance evaluation

5.1. Simulation environment

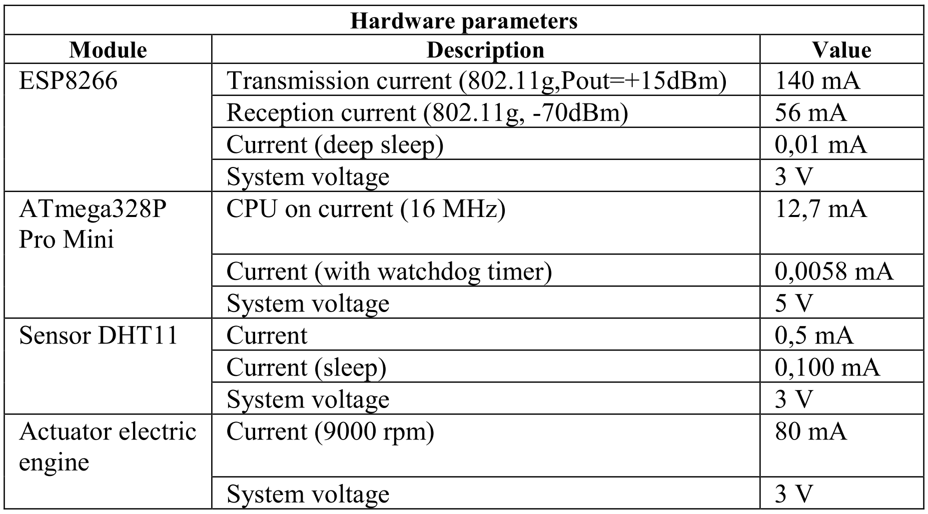
The authors propose an event driven simulator for the evaluation of the system. The simulator is implemented entirely in Java language and uses the bcprov-jdk15on-160 library for ECC. Fig. 12 reports a diagram containing the main modules. The environment module creates, initializes and maintains all nodes. The fog node uses an MQTT broker for the management of topics, an authentication manager for managing stored credentials, generating and validating session tokens, a key manager for managing keys exchange and maintaining lightweight nodes public keys and a Host IDS module for validating packets timestamps. Lightweight nodes send (sensor nodes) or receive (actuator nodes) packets periodically.

Each lightweight node has a Host IDS for the validation of packets timestamps; it changes its key pair periodically and sends the public key to the broker. Similarly, the broker changes its key pairs periodically and sends them to the related lightweight nodes with an active MQTT session. The simulator provides several editable context parameters included in a formatted CSV file. During the simulation, some data are collected and, at the end of the simulation, these data are stored on a formatted CSV file from which it is possible to generate some graphs.

5.2. Lightweight nodes power consumption

A lightweight node implements a state finite automaton. Each lightweight node has a battery with a specified capacity. The lightweight node power consumption considers various modules: CPU, WiFi module, sensor module, actuator module. For each module, the energy consumption is calculated as:

$$E = T * I * V$$



Hardware parameters		
Module	**Description**	**Value**
ESP8266	Transmission current (802.11g,Pout=+15dBm)	140 mA
	Reception current (802.11g, -70dBm)	56 mA
	Current (deep sleep)	0,01 mA
	System voltage	3 V
ATmega328P Pro Mini	CPU on current (16 MHz)	12,7 mA
	Current (with watchdog timer)	0,0058 mA
	System voltage	5 V
Sensor DHT11	Current	0,5 mA
	Current (sleep)	0,100 mA
	System voltage	3 V
Actuator electric engine	Current (9000 rpm)	80 mA
	System voltage	3 V

Fig. 13. Hardware parameters.

where E is the power consumption in Wh, T is the usage time, I is the needed current and V is the system voltage. For each module it is considered the power consumption for activation periods and the power consumption for sleep periods. For WiFi module it is also considered the power consumption for transmitting packets. The hardware values used in the simulations are reported in Fig. 13.

5.3. *Main default parameters*

The following table in Fig. 14 contains the main default parameters used in the simulations.

5.4. *Energy consumption*

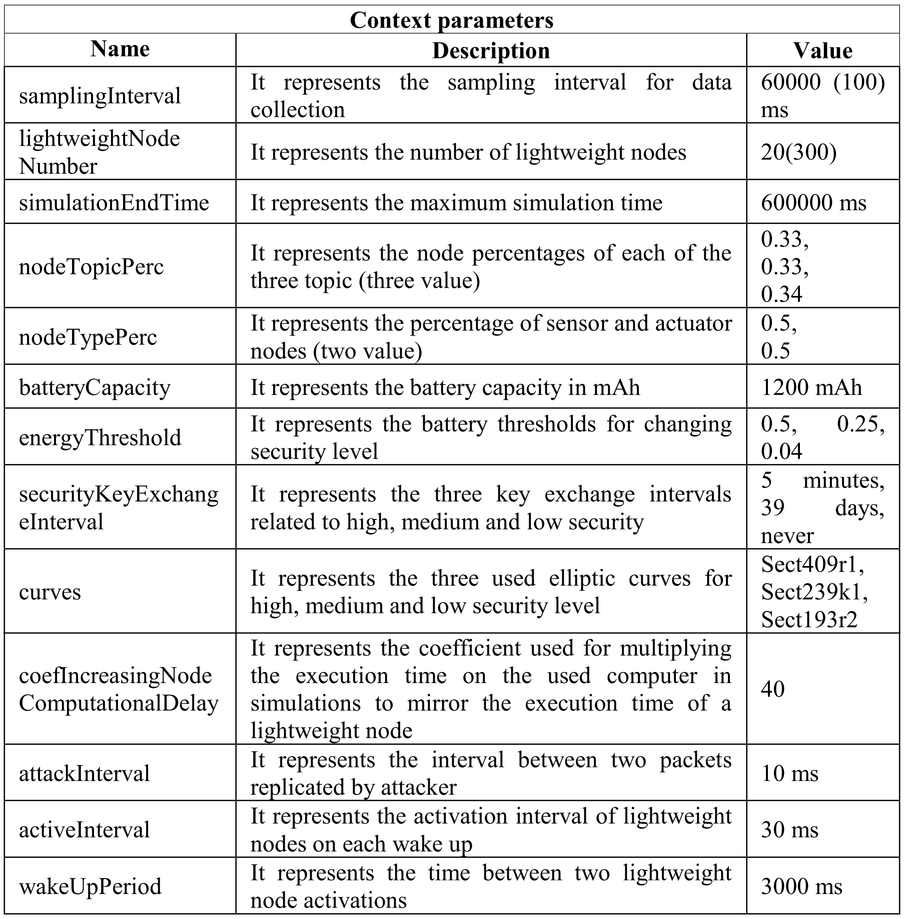
The graphs in Figs. 15 and 16 show an average power consumption repartition for nodes using the three different types of topic and the three different security levels. The power consumption is expressed in Wh and is divided, for each type of node, among: CPU module, WiFi module, Sensor module and Actuator module. The graph shows that the average power consumption of nodes using a ciphered topic are greater than non-ciphered one. This difference comes from a high CPU usage, which depends on the encryption of the exchanged payloads. The average power consumption of nodes using the Authenticated_Cipher_topic is greater than those of the Cipher_topic as, in that topic, nodes also manage authentication. It is possible to observe that the higher is the security level, the higher is the CPU usage. In the presence of cryptography, the CPU power consumption becomes greater than that of WiFi because of its high use time. In the presence of cryptography the CPU power consumption becomes the principal cause of energy use. This is the challenge which this work seeks to address. In fact, we propose reducing this CPU energy consumption by increasing the overhead caused by exchanged keys, given that the WiFi power consumption is lower than the CPU one.

5.5. *Dynamic security*

The graphs in Figs. 17 and 18 show that by using a dynamic security mechanism it is possible to increase the system lifetime but the lower the security level, the higher is the generated overhead related to the key exchange. This overhead is contained compared to the data exchange related to the system function. In fact, in the worst case we use a low security level which consists of sending a key packet every 5 min while sensor nodes send messages every few seconds. These results are related to the fact that a higher security level needs a higher CPU usage and then a higher energy, and this reduces the system lifetime. Moreover, by decreasing the security level it is necessary to increase the key change frequency and this causes a higher overhead (key exchange). A good solution must consider the tradeoff between the overhead caused by the key change frequency and the consumed energy by encryption process.

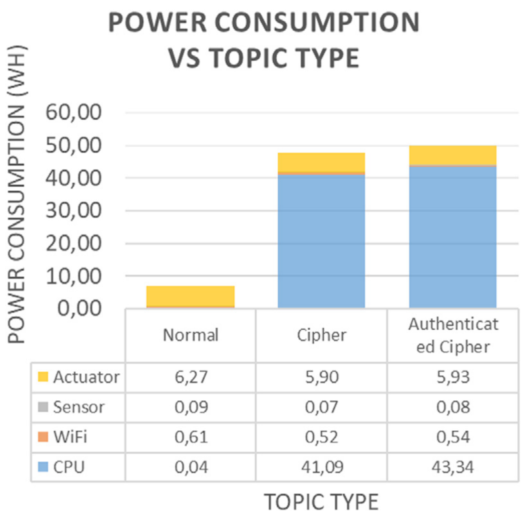
5.6. *ActiveInterval tuning*

The *activeInterval* represents the time interval for which a lightweight node stays active after a wake up and before going sleep. An ack timeout occurs when a lightweight node wakes up, sends a message but the relative ack does not reach it before going to sleep. The graph in Fig. 19 shows that the ack timeout number of lightweight nodes remains low



Context parameters		
Name	**Description**	**Value**
samplingInterval	It represents the sampling interval for data collection	60000 (100) ms
lightweightNode Number	It represents the number of lightweight nodes	20(300)
simulationEndTime	It represents the maximum simulation time	600000 ms
nodeTopicPerc	It represents the node percentages of each of the three topic (three value)	0.33, 0.33, 0.34
nodeTypePerc	It represents the percentage of sensor and actuator nodes (two value)	0.5, 0.5
batteryCapacity	It represents the battery capacity in mAh	1200 mAh
energyThreshold	It represents the battery thresholds for changing security level	0.5, 0.25, 0.04
securityKeyExchang eInterval	It represents the three key exchange intervals related to high, medium and low security	5 minutes, 39 days, never
curves	It represents the three used elliptic curves for high, medium and low security level	Sect409r1, Sect239k1, Sect193r2
coefIncreasingNode ComputationalDelay	It represents the coefficient used for multiplying the execution time on the used computer in simulations to mirror the execution time of a lightweight node	40
attackInterval	It represents the interval between two packets replicated by attacker	10 ms
activeInterval	It represents the activation interval of lightweight nodes on each wake up	30 ms
wakeUpPeriod	It represents the time between two lightweight node activations	3000 ms

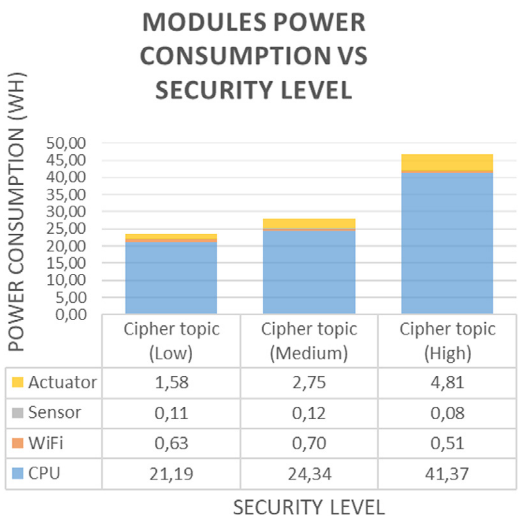
Fig. 14. Main context parameters.



	Normal	Cipher	Authenticated Cipher
🟨 Actuator	6,27	5,90	5,93
⬜ Sensor	0,09	0,07	0,08
🟧 WiFi	0,61	0,52	0,54
🟦 CPU	0,04	41,09	43,34

Fig. 15. Power consumption vs. topic type.

when *activeInterval* remains higher or equal than 30 ms. When the *activeInterval* value becomes lower than 20 ms, the number of ack timeout increases rapidly. This means that the Round Trip Time between a lightweight node and the fog node is higher than 20 ms, then a lightweight node must stay active for a time greater than 20 ms for receiving the ack.



	Cipher topic (Low)	Cipher topic (Medium)	Cipher topic (High)
🟨 Actuator	1,58	2,75	4,81
⬜ Sensor	0,11	0,12	0,08
🟧 WiFi	0,63	0,70	0,51
🟦 CPU	21,19	24,34	41,37

Fig. 16. Modules power consumption vs. security level.

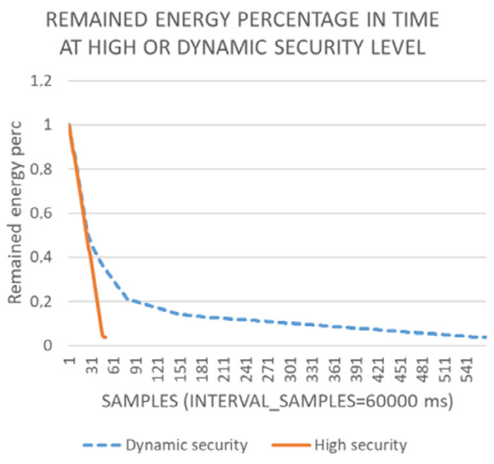


Fig. 17. Remained energy percentage in time at high or dynamic security level.

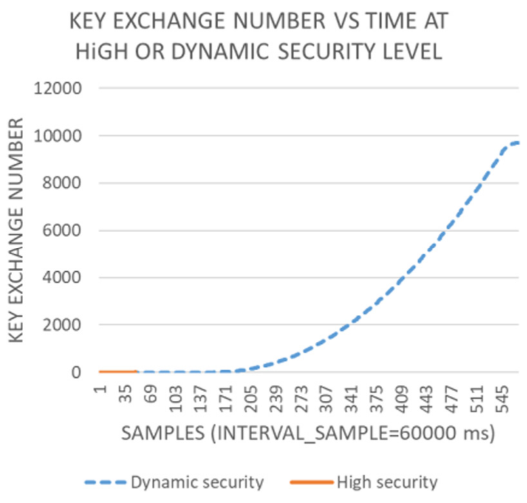


Fig. 18. Key exchange number vs. time at high or dynamic security level.

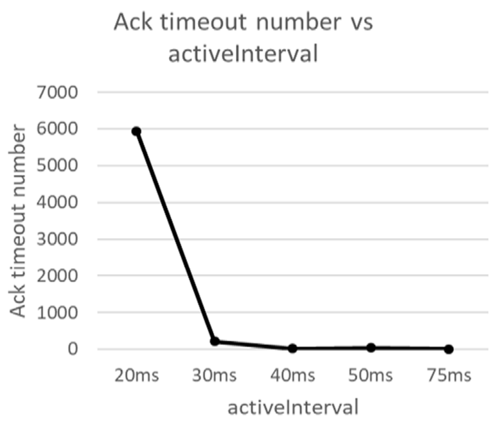


Fig. 19. Ack timeout number vs. activeInterval.

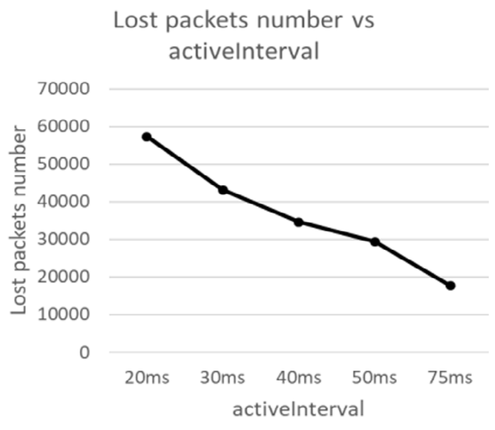


Fig. 20. Lost packets number vs. activeInterval.

The ack timeout number must be minimized for a better system functioning. We consider the replicated packets from the attacker node that do not reach the destination because it is in sleep mode at reception time as lost. The graph in Fig. 20 shows that the lost packets number decreases at the increasing of *activeInterval*. Lower is the active time of a lightweight node and lower is the number of received replicated packet. This happens because when a node is in sleep mode, it cannot receive packets. The lost packets number must stay high to reduce energy waste by the lightweight nodes in the dropping process of replicated packets. The graph in Fig. 21 shows that the dropped malformed packets number increases at the increasing of *activeInterval* because the lightweight nodes remain active for much time and so they receive more replicated packets. The dropped malformed packets number must be minimized to reduce the energy waste which is relative to the dropping process of replicated packets. This value is linked to the lost packet number because higher is the number of the lost packet and lower is the number of replicated packets reaching the lightweight nodes that needs to be dropped as malformed. A good tradeoff for the *activeInterval* parameter is 30 ms.

5.7. Replay attacks

The graph in Fig. 22 is relative to a Replay attack to the lightweight nodes, which has the purpose of causing malfunctions and energy waste in the lightweight nodes; it shows that by using an *activeInterval* of 30 ms it is possible to lose almost all replicated packets. This permits a decrease of the energy waste relative to the dropping process. Many received replicated packets are dropped by the destination as malformed packets because they contain an old timestamp. The received replicated packets relative to Normal_topic are accepted because they are not protected by Replay attacks. In this context the lightweight nodes using the Normal topic represent 33% of the total. This graph shows that by using the proposed technique we can preserve lightweight nodes energy waste by minimizing number of replicated packets by attacker that reach the destination. Furthermore, the packets that reach the destination are dropped as malformed. In the Replay attack to the fog node all packets are received by the destination because of the fog node does not go in sleep mode. The purpose is to cause system malfunctions in the authenticated topics for which the attacker does not know

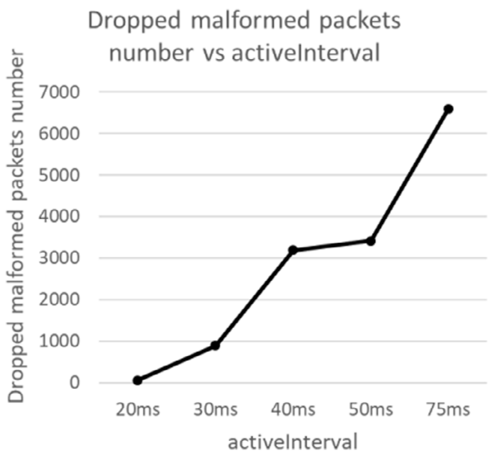


Fig. 21. Dropped malformed packets number vs. activeInterval.

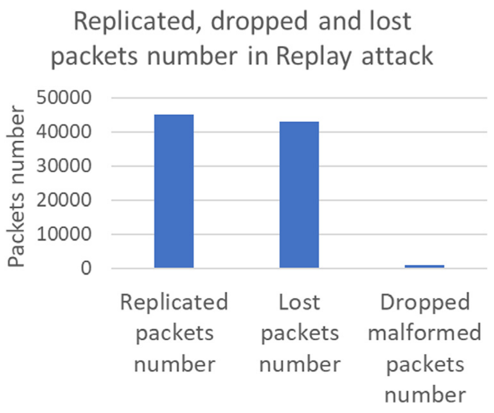


Fig. 22. Replay attack to lightweight nodes.

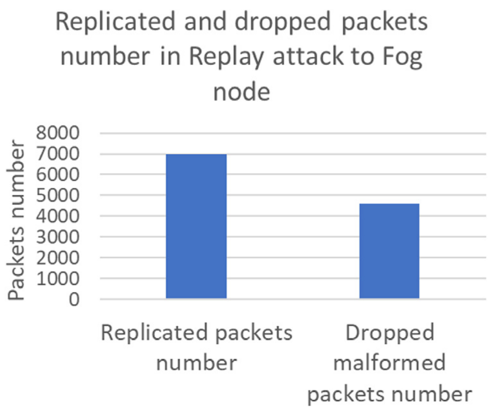


Fig. 23. Replay attack to fog node.

the credentials. The graph in Fig. 23 shows that in the Replay attack to the fog node many received replicated packets are dropped by the destinations as malformed packets because they contain an old timestamp. The received replicated packets relative to Normal_topic (about 33%) are accepted because they are not protected by Replay attacks. This means that topics using the proposed mechanism are able to reject the replicated packets in a Replay attack and this can reduce system malfunctions.

6. Conclusions

This work proposes a new dynamic IoT security system in which there are sensor and actuator nodes exchanging data through a secured MQTT protocol in a fog network. The proposal encrypts the MQTT payloads with ECC for mitigating data tampering and eavesdropping and adds a timestamp to the payloads and uses lightweight node wake-up patterns for mitigating Replay attacks. To reduce the encryption energy consumption, we have proposed to change dynamically the key-strength of the used ECC on the basis of the residual energy capacity, being careful to adapt the key change frequency on the basis of the current key-strength. Moreover, the use of a wake-up pattern decreases the received replicated packets and the consumed energy to drop them. The proposed security system was validated by the implementation of an event driven simulator, from which the generated graphs had shown that the proposal increases the system lifetime(paying with an increase of the overhead related to keys exchange) and mitigates Replay attacks reducing the energy waste related to the dropping process. Our future work will mainly focus on the application of new elliptic curves and others cryptography techniques to both fog and cloud computing context with the aim to reduce further the consumed energy. Moreover we are already working on the mitigation of DoS attacks in fog computing.

Declaration of competing interest

None declared under financial, general, and institutional competing interests.

Acknowledgment

This work was supported by 'POR Calabria FSE/FESR 2014/2020 – International mobility of Ph.D. students and Research Grants/Type A Researchers" – Actions 10.5.6 and 10.5.12 actuated by 'Regione Calabria, Italy.

References

[1] E.D. Kavyashree, H.D. Vidyashree, H. Anil Kumar, A survey of internet of things (IoT)-applications, merits, demerits & challenges, in: IJIRCCE, Vol. 6, No. 2, 2018.
[2] P. Parhana, M. Lakshmaiah, S. Allaudheen, S. Dastagiri, M. VijayaSaritha, Review on internet of things: Recent applications and its challenges, in: IJAREEIE, 2017, pp. 6–11.
[3] N. Abbas, et al., Mobile edge computing: A survey, IEEE Internet Things 5 (1) (2018) 450–465.
[4] E. Ahmed, M.H. Rehmani, Mobile Edge Computing: Opportunities, solutions, and challenges, Future Gener. Comput. Syst. (2017).
[5] Hany F. Atlam, Robert J. Walters, Gary B. Wills, Fog computing and the internet of things: A review, Big Data Cogn Comput (2018).
[6] D. He, S. Chan, M. Guizani, Security in the internet of things supported by mobile edge computing, multiple access mobile edge computing for heterogeneous IoT, IEEE Commun. Mag. (2018).
[7] M. Mukherjee, et al., Security and Privacy in Fog Computing: Challenges, IEEE, 2017.
[8] Y. Saleem, F. Salim, M.H. Rehmani, Resource management in mobile sink based wireless sensor networks through cloud computing, in: Resource Management in Mobile Computing Environments, in: Springer-Verlag Handbook, vol. 3, Springer, 2014, pp. 439–459.
[9] E. Ahmed, A. Gani, M. Sookhak, S.H. Ab Hamid, F. Xia, Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges, J. Netw. Comput. Appl. 52 (2015) 52–68.
[10] J. Shuja, et al., Case of arm emulation optimization for offloading mechanisms in mobile cloud computing, Future Gener. Comput. Syst. (2016).
[11] L. Valerio, M. Conti, A. Passarella, Energy efficient distributed analytics at the edge of the network for IoT environments, Perv. Mobile Comput. 51 (2018) 27–42.
[12] T.K. Meng, et al., A secure and cost-efficient offloading policy for Mobile Cloud Computing against timing attacks, Perv. Mobile Comput. 45 (2018) 4–18.
[13] K.-H. Wang, C.-M. Chen, W. Fang, T.-Y. Wu, A Secure authentication scheme for Internet of Things, Perv. Mobile Comput. 42 (2017) 15–26.
[14] S. Kalra, S.K. Sood, Secure authentication scheme for IoT and Cloud servers, Perv. Mobile Comput. 24 (2015) 210–223.
[15] A. Mektoubi, H.L. Hassani, H. Belhadaoui, M. Rifi, New approach for securing communication over MQTT protocol A comparison between RSA and Elliptic Curve, in: 3rd Int. Conf. on Systems of Collaboration (SysCo), 2016.
[16] A.A. Diro, N. Chilamkurti, N. Kumar, Lightweight Cybersecurity Schemes using Elliptic Curve Cryptography in Publish-Subscribe Fog Computing, Springer Science + Business Media New York, 2017.
[17] Meena Singh, et al., Secure MQTT for Internet of Things (IoT), in: Fifth International Conference on Communication Systems and Network Technologies (CSNT), 2015, Gwalior, India.
[18] I. Chatzigiannakis, et al., A privacy-preserving smart parking system using an IoT elliptic curve based security platform, Comput. Commun. 89 (2016) 165–177.
[19] N. Park, N. Kang, Mutual authentication scheme in secure internet of things technology for comfortable lifestyle, Sensors (2016).
[20] S. Sciancalepore, A. Capossele, G. Piro, G. Boggia, G. Bianchi, Key Management Protocol with Implicit Certificates for IoT Systems, ACM, IoT-Sys, 2015.
[21] M. Hammi, et al., A lightweight iot security protocol, in: 1st CSNetConference, Rio de Janeiro, Brazil, 2017.
[22] Paavan Rughoobur, Leckraj Nagowah, A lightweight replay attack detection framework for battery dependent IoT devices designed for healthcare, in: Int. Conf.Infocom Technologies and Unmanned Systems, 2017.
[23] A. Vanathi, B. Sowjanya Rani, Cloning attack authenticator in wireless sensor networks, IJCST 3 (1) (2012) Spl. 5.
[24] A. Lupia, F. De Rango, Performance evaluation of secure AODV with trust management under an energy aware perspective, in: SPECTS, 2014.
[25] F. De Rango, S. Marano, Trust-based SAODV protocol with intrusion detection and incentive cooperation in MANET, in: Proc. of the IWCMC, 2009.
[26] F. De Rango, D.C. Lentini, S. Marano, Static and dynamic 4-way handshake solutions to avoid denial of service attack in Wi-Fi protected access and IEEE 802.11i, EURASIP J. Wireless Commun. Networking 1 (2006) 047453.
[27] S. Barber, P. Mahalle, A. Stango, N. Prasad, Proposed security model and threat taxonomy for the internet of things, in: Recent Trends in Network Security and Applications, Springer Berlin Heidelberg, 2010, pp. 420–429.

[28] P.R. Egli, MQTT MQ Telemetry Transport – An Introduction to MQTT, A Protocol for M2M and IoT Applications, indigoo, 2016.
[29] Y. Yang, et al., A survey on security and privacy issues in internet-of-things, IEEE Internet Things J. 4 (5) (2017) 1250–1258.
[30] J.A. Amalraj, Jose, J.J. Raybin, A survey paper on cryptography techniques, Int. J. Comput. Sci. Mobile Comput. 5 (2016) 55–59.
[31] E. Rescorla, SSL and TLS: Designing and Building Secure Systems, Addison-Wesley Reading, 2001.
[32] N. Koblitz, Elliptic curve cryptosystems, Math. Comp. 48 (1987) 203–209.
[33] D. Hankerson, et al., Guide to Elliptic Curve Cryptography.
[34] V.S. Miller, Use of Elliptic Curves in Cryptography, Springer-Verlag Berlin Heidelberg, 1986.
[35] K. Sumanth, M. Jayabhaskar, Secure digital signature scheme based on elliptic curves for internet of things, IJECE 6 (3) (2016) 1002–1010.
[36] C. Research, SEC 2: Recommended Elliptic Curve Domain Parameters, Standards for Efficient Cryptography (SEC), 2000.
[37] BlueKrypt, Cryptographic key length recommendation, 2018, https://www.keylength.com/en/4/.
[38] The Legion of the Bouncy Castle, Available: https://www.bouncycastle.org/latest_releases.htm.