Securing Communication between Fog Computing and IoT Using Constrained Application Protocol (CoAP): A Survey

Fahd A. Alhaidari[1] and Ebtesam J. Alqahtani[2]
[1] Department of Computer Information System, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam, 31441, Saudi Arabia
[2] Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, P.O. Box 1982, Dammam, 31441, Saudi Arabia.
Email: faalhaidari@iau.edu.sa; ebtesam.j.alqahtani@gmail.com

Abstract—Nowadays, cloud computing and IoT devices are widely used and involved in our life. However, the current cloud computing paradigm still have some limitation mainly related to the latency, location, and mobility. Thus, to overcome such limitations, Fog Computing was introduced as an intermediate layer between the IoT devices and cloud computing to providing multiple benefits such as low latency and mobility. However, the security of Fog Computing protocols is still a concern especially those related CoAP protocol. CoAP protocol is still does not have the reliable standards for securing its architecture and there is a huge lack of the main researches on how security can be managed or provided to CoAP. Hence, this paper surveys the CoAP protocol, its architecture, security and different proposed techniques to secure CoAP protocol. The paper provides a solid reference for the Fog Computing and CoAP protocol security as well as it proposed a taxonomy for the literature review to ease understanding all available techniques to secure CoAP.

Index Terms—Fog computing, IoT, CoAP, security, DTLS, wireless sensor networks

I. Introduction

Currently cloud computing and IoT devices are widely used and it has significantly gotten more involved in our life. However, the current cloud computing paradigm still have some limitation, and it can't satisfy the requirements of supporting mobility, location and low latency. Unacceptable latency may prevent some critical IoT applications and services such as disaster management applications of benefiting from cloud computing due to inherent problems [1]. The critical issue, is that as the size of cloud is growing, the network latency will increase as well which is not acceptable for critical IoT applications [2]. Thus, the Fog Computing concept has been introduced to address the cloud computing limitations. Fog Computing is an extension of the cloud computing, it has multiple benefits such as low latency, location awareness, and mobility. Also, it is an appropriate paradigm for many IoT services [3].

The introduction of Fog Computing requires application protocols to ensure the interaction and communication among the involved devices and applications. Among these application protocols are: Constrained Application Protocol (CoAP) [4], [5], Message Queue Telemetry Transport (MQTT) [6], Advanced Message Queuing Protocol (AMQP) [7], Data Distribution Services (DDS) [8], ZigBee [9], UPnP [10], DPWS [11]).

However, CoAP is the most popular protocol used in IoT application due to its features such as: its simplicity when being used by developers, its lightweight style in terms of power consumptions and communication, mobility, portability, and having enough techniques to improve the data security and integrity [12]-[14]. Md. Motaharul *et al*. [15] proposed cloud based architecture for harmonizing IoT using CoAP protocol and their evaluation experiments showed that CoAP protocol is more suitable than MQTT in implementing the communication between sensors in cloud computing environment. Moreover, CoAP has rapidly gained the adoption and supports by large companies [16] and it has been introduced by researchers, due to its lightweight property, to be used in several domains ranging from smart homes to the industrial WSNs [17]-[22].

The security of Fog Computing protocols especially CoAP protocol is an important topic as it does not have the reliable standards for secure architectures. In CoAP, Datagram Transport Layer Security (DTLS) [23] is proposed to be implemented on top of UDP to secure its messages. However, DTLS does not suit the CoAP proxy and it was not designed initially for resource constrained devices; for example, in order to perform the handshake process DTLS need six flight messages which increases the communication overhead and consume constrained devices energy. DTLS has been studied by different researchers to find out how it secures CoAP communication messages. However, the main challenge which face CoAP is the lack of the key researches on how security can be managed and implemented [24], [25]. Thus, this paper surveys CoAP protocol, its architecture, security and security enhancement suggestions.

Fog Computing is not a replacement to the cloud computing; however, it is considered as an extension or intermediate layer between cloud computing and IoT devices. The term Fog Computing was introduced by

Manuscript received April 5, 2019; revised December 5, 2019.
Corresponding author email: faalhaidari@iau.eduu.sa.
doi:10.12720/jcm.15.1.14-30

Cisco Systems as a bridge between cloud computing and Internet of Things (IoT) devices to overcome the gap [26]. Using Fog Computing, the cloud services can be extended closer to IoT devices. It is a highly virtualized platform that provides different services such as storage, and networking services [27].

Fog Computing offers flexible, inexpensive and portable in terms of both hardware and software [28]. It also provides location awareness, mobility of things, and bandwidth requirements for IoT devices. Moreover, it provides a low latency advantage by allowing processing functions to take place at the network edge [1] [29], [30]. Fog Computing have relatively small computing resources such as memory, processing and storage comparing to the cloud computing. However, it is able to process the data generated from diverse devices, and it can be installed on low specification devices such as switches [28].

According to [31], Fog Computing extends the capability of cloud by introducing its four layers. Fig. 1 shows the Fog Computing architecture.

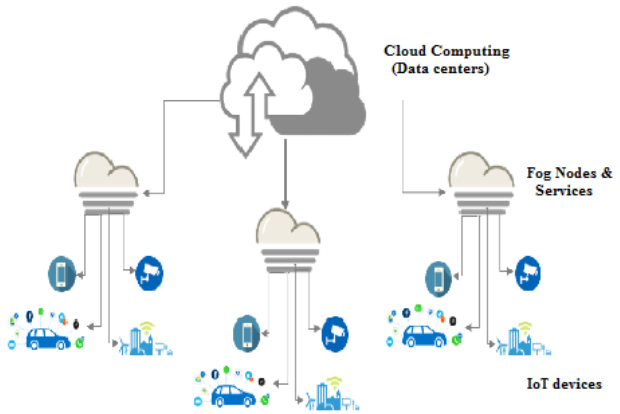

Fig. 1. Fog computing architecture [31].

As illustrated, the data centre layer provides the fog nodes with large volume of data, resources and services to the fog node. The second and third layers are the fog nodes and services, which provide the IoT devices with needed services and data, that reduces the latency and gaps between cloud and IoT devices. The fourth layer is IoT devices such as sensors, embedded systems, etc. [31].

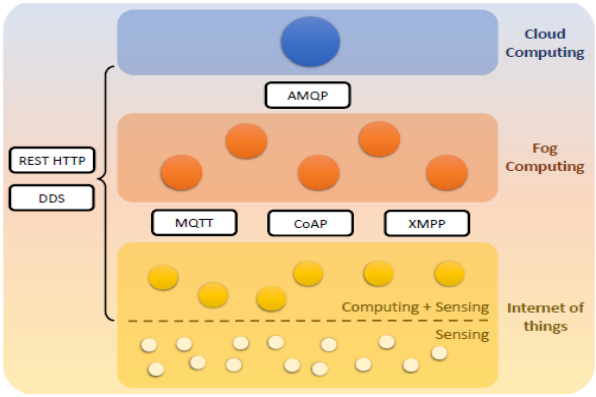

Fig. 2. Application protocols in IoT, Fog and cloud networks [32].

Fog Computing is an extension of the cloud computing, it has multiple benefits such as low latency, location awareness, and mobility. Also, it is an appropriate paradigm for many IoT services. The main protocols that have been considered for communication between cloud and fog, and Fog and IoT devices are illustrated in Fig. 2. The communication between Fog Computing and IoT devices are carried out using CoAP, MQTT, and XMPP protocols [32].

The rest of the paper is organized as follows: Section 2 presents the background of the related aspects of Fog Computing, In Section3, a literature review is presented. Section 4 shows the discussion and analysis of the related reviewed studies. Findings and recommendations are presented in Section 5 and finely the conclusion is given in Section 6.

II. Background

This section introduces a background on CoAP as an application layer protocol designed for resource-constrained devices.

CoAP is adopting the User Datagram Protocol (UDP) protocol to be adopted by low-bandwidth connections and low-computational-power devices [33], [34]. Unlike HTTP, CoAP supports multicast. CoAP uses Efficient XML Interchanges (EXI) format which consume less space compared to XML/HTML binary format [35]. In general, Fig. 3 shows the comparison between HTTP and CoAP protocols.

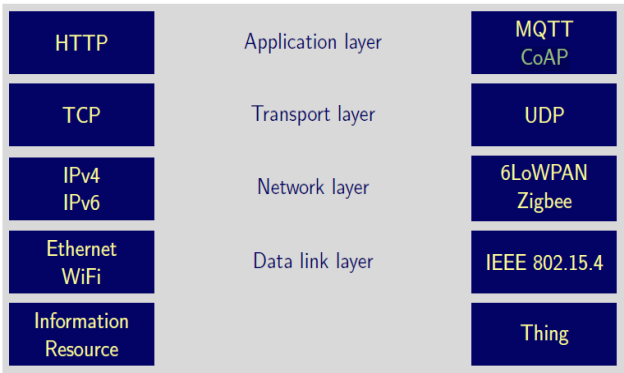

Fig. 3. HTTP and CoAP comparison [36].

A. CoAP Architecture

CoAP is a stateless protocol relying on the client-server architecture with the usage of request- response model for exchanging messages. Like HTTP, CoAP uses Representational State Transfer (REST) model, where it addresses the server resources by Uniform Resource Identifier (URI) that can be accessed from a client by sending a request (GET, POST, PUT and DELETE) to the server referring to such URLs [34].

CoAP includes two layers which are messaging layer and request/ response layer. The message layer responsible about the redundancy and consistency of any message, while the request/response layer is responsible about the connectivity and communication [35].

There are four types of messages which is supported by CoAP which are [35]:

Confirmable Messages (CON): which are all messages marked as confirmable messages (reliable messaging mode). CON messages are sent out based on the default timeout and exponential back-off mechanism, until receiving the acknowledgment message (ACK) that must hold the same message ID sent by the sender. If the reply was not successful, the recipient will send a reset message (RST) to reset the communication. Fig. 4 shows the CON and ACK messages between the client and the server.

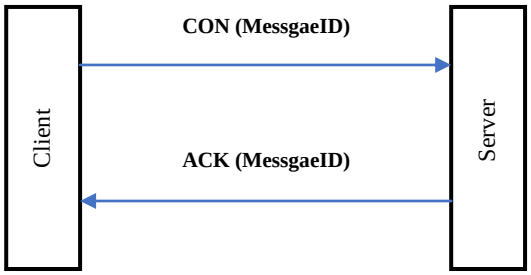

Fig. 4. Confirmable messages communication

Non-Confirmable Messages (NON): which are all messages marked as not confirmable messages (unreliable messaging mode). The recipient is able only to send reset message if it is unable to respond to the NON message. Fig. 5 shows the NON-message between the client and the server.

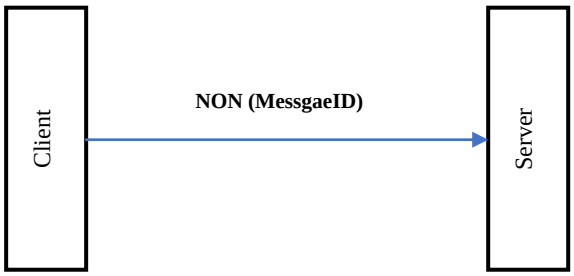

Fig. 5. Non-confirmable messages communication.

Acknowledgment Messages (ACK): which is the message sent back to the sender by the recipient in the reliable messaging mode. It must contain the same message ID sent in the CON message by the sender as shown in Fig. 4.

Reset Messages (RST): Which is the message sent back to the sender by the recipient to reset the communication between the client and the server.

Messages in request/response layer can be described as followings [35]-[37]:

Piggy-Backed Message: Which is sent immediately by the server after receiving CON or NON messages. In general, the message is called ACK message. Fig. 6 and Fig. 7 show the successful and failure responses which contain sucess and failure codes respectively.

Separate Message: Which is an empty message sent by the server to stop the client from resending the message. In general, it is sent when the server can't respond to the client immediately and once the server become ready, an ACK message will be sent. This type of message takes some time to be delivered. Fig. 8 illustrates the separate message communication.

Non-Confirmable Message: When a non-confirmable message is sent by the client, the server could respond by NON or CON messages as shown in Fig. 9.

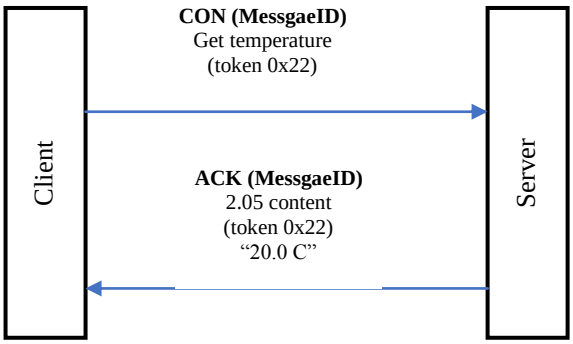

Fig. 6. The successful response results of GET method.

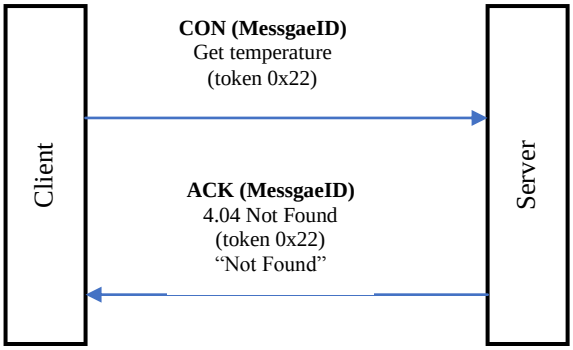

Fig. 6. The failure response results of GET method

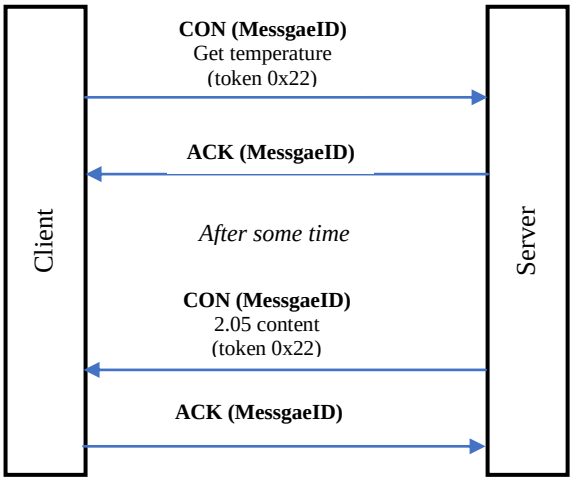

Fig. 7. Get request with a separate response.

CoAP message follows RESTful architecture making it suitable to be used on constrained devices that required lightweight protocols. The messages have a simple binary format of a 4-byte header that holds the required information of each message including the payload options as shown in Fig. 10, where,

- *Version (V)* holds to CoAP version number.
- *Type (T)* holds the message type.
- *Token Length (TKL)* holds the length of the token.

- *Code* holds the request or response code.
- *Message ID* is used to identify the message for the purpose of matching and detecting the redundancy.
- *Token might* be used as an association for all requests and responses.
- *Options* are attached at the end of a message with the possibility to have several options and can hold a payload.
- *Payload* is mainly added to the end of the UDP datagram and its value determined based on the size of the datagram [35].

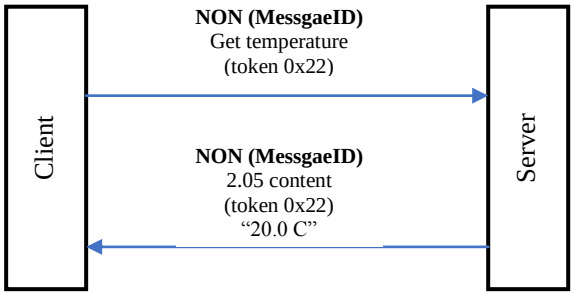

Fig. 8. Non-confirmable request and response.

Version (V)	Type (T)	Token Length (TKL)	Code	Message ID
Token (If any)				
Options (If any)				
Payload (If any)				

Fig. 9. CoAP message header [35]

B. *Security in CoAP*

The security of CoAP is an important aspect due to not having the reliable standards to secure CoAP architecture. Thus, Datagram Transport Layer Security (DTLS) has been proposed to be used on top of UDP to secure its messages communications. DTLS provides the security services including confidentiality, integrity, authentication, and non-repudiation services utilizing the fundamentals of AES/CCM [24].

According to [25], [35], DTLS manages crucial security factors such as authentication, confidentiality, key management, and data integrity of the messages. DTLS has four modes of security which is used by CoAP with several applications as described below:

NoSec Mode: Where it assumes that another protocol layer will implement the security mechanism, and hence messages are transfered with no security.

PreSharedKey Mode: Where the devices that are allowed to use the same system are already programmed with single symmetric key which help in communication with other devices.

RawPublicKey Mode: it is considered as an essential in the implementation of CoAP and generally is adopted by the devices that required authentication, which uses asymmetric key for each device to help in identifying these devices and interact with them.

Certificates Mode: it is considered as an authentication technique for devices that implement CoAP with an X.509 certificate.

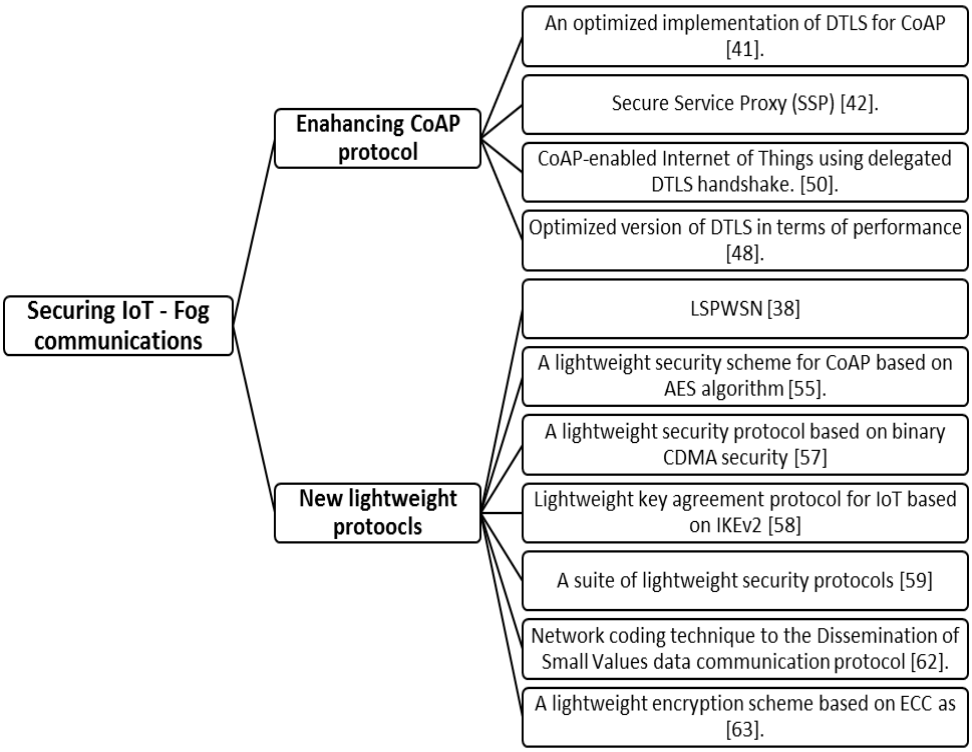

Fig. 10. Taxonomy of the literature review

DTLS proposed by some researchers to be be implemented in CoAP to provide the security. However, applying DTLS has some limitations such as: large message and handshake process which uses six flight messages to complete the handshake process which will increase the number of the transmitted bytes, and consumed energy. Also, the main importamt limitation is that, DTLS was not designed for resource constrained devices; and it uses [38].

DTLS has been studied by some researchers to find out how does DTLS applied on CoAP to provide security. However, there is a huge lack of the main researches on how security can be managed or provided to CoAP [24] [38]. The biggest challenges which face CoAP are the lack of the important researches related to the security aspects along with keeping the high performance after providing the needed protection to the communications.

III. Literature Review

CoAP security mechanisms have been studied by different researchers to find out how the CoAP communications are secured against different types of attacks. The main researches in this filed have been surveyed; however, there is a major lack of the researches which focused on CoAP and its security. Accordingly, in this section the key researches which focused on CoAP security analysis, improvements and weaknesses have been surveyed. After that different lightweight protocols which are suitable for limited resources devices have been reviewed.

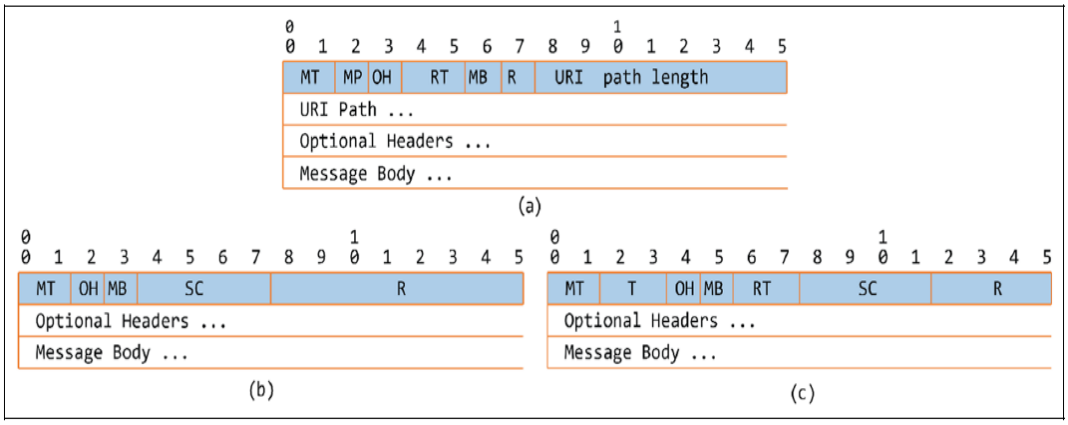

Fig. 11. LSPWSN Messages [38].

The literature in general is divided into two main categories which is either enhancing CoAP security or proposing a new lightweight protocol to secure the communications between IoT devices and fog nodes. Fig. 11 shows the proposed taxonomy of the main studied literature review. Other studies have been explored in the following subsections.

A. Lightweight and Secure Protocol for Wireless Sensor Networks (LSPWSN)

Figueroa P. *et al.* in [38] presented a lightweight, efficient and secure protocol named Lightweight and Secure Protocol for Wireless Sensor Networks (LSPWSN) to be used for the Web Services in wireless sensors. It is proposed to work over TCP and IPv6 protocols, using the 6LoWPAN standard. Another study presented in [39] introduced a compression method on IPSec protocol headers to provide more flexibility and extendibility to 6LoWPAN. Also, R. Garg and S Sharma in [40] presented a new compression method for IPv6 header in 6LoWPAN environment providing more efficiency of the protocol.

Web services can't be easily available to the limited resources such as sensor nodes. Thus, LSPWSN is designed to overcome this issue and to provide secure web services for limited resources. In general, it follows the RESTful approach and it uses binary encoding to encode headers and payload. It uses three messages types which are request, response and publication messages. Fig. 12 illustrates the three messages where (a) is the request message, (b) is the response message and (c) is the publication message.

LSPWSN is considered as point-to-point protocol, end-to-end protocol, stateless protocol, and secure protocol. It uses a lightweight stream cipher to provide needed security. Using the lightweight stream cipher is more efficient than using DTLS or IPsec to secure CoAP in terms of performance and complexity. According to the authors, IPSec cannot be applied directly in WSN because it does not provide neither data aggregation nor in-network processing. Nevertheless, LSPWSN protocol proposes using a lightweight stream cipher called SNOW version 2 to secure the communication [38].

In order to evaluate the performance of LSPWSN and CoAP an experiment have been carried out and executed using on "Zolertia Z1" platform which is emulated through Contiki Cooja Network Simulator. The main goal of the experiment is to prove that LSPWSN is energy efficient even when using TCP as transport layer. LSPWSN has been evaluated via a quantitative analysis which is carried out using four metrics: memory footprint, transferred bytes per client-server transaction, service response time, and energy consumption [38].

The results showed that LSPWSN RAM and ROM consumptions are better than CoAP by 11.37% and 4.27% respectively. In terms of the bandwidth usage or the transferred bytes per client-server transaction, the LSPWSN transaction is larger than CoAP by nearly five times, and it needs nearly four times more packets than CoAP. These bytes and packets are required to provide required security and reliability. Evaluating the service response time has been executed through 1000 requests. The results showed that the average service response time of all 1000 requests is 0.95 seconds with LSPWSN and 0.47 seconds with CoAP. Lastly, the average total energy consumption of the 1000 requests for LSPWSN and CoAP 1000 is 0.478 and 0.446 mW respectively. Since LSPWSN uses TCP, the LSPWSN web server handle seven more packets than the CoAP web server does and this increases the service response time as well as the energy consumption. In general, LSPWSN protocol presents a more stable behaviour than CoAP. Since CoAP uses UDP, there is no guarantee of message delivery [38].

B. Optimized Implementation of DTLS for CoAP

In [41], Capossele A. et al. have developed an optimized implementation of DTLS for CoAP, by integrating DTLS protocol inside the CoAP and combining existing and Elliptic Curve Cryptography (ECC) optimizations, as well as minimizing ROM utilization. DTLS has been optimized by combining the connection-oriented communication and the fragmentation. The connection-oriented communication is provided by CoAP message layer while the fragmentation is obtainable using block-wise transfer feature provided by CoAP. Combining both mechanisms guarantees that DTLS is compliant to the security standard as well as lighter than the standard implementation [41].

Moreover, the authors developed a RESTful DTLS connection to be a CoAP resource to allow large re-use of CoAP functionalities and code, as well as providing CoAP with the abilities to optimize the resources usage. This connection is created once a new secure session is requested by the client. To avoid consuming server resources in case of DoS attack, a stateless cookie technique has been applied where all the clients are enforced to re-transmit the Client Hello message with the attached cookie and based on the cookie validation, the server will decide whether to continue the handshake process or not [41].

In general, the proposed method ensures the communication reliability by CON and ACK messages and consequently, a new DTLS session is created on the server. The Client Hello messages are sent respectively by the server and the client to increase the security and mitigate the DoS attack. The proposed DTLS enhancement utilizes the efficiency of CoAP block-wise transmission to perform all the fragmentation tasks. When both client and server receive Finished messages, the secure session is established [41].

To demonstrate the viability of their solution, the authors applied their proposed solution on the MagoNode and compared it to the standard implementation of DTLS. The experiments results show that the optimized ECC solution outperform the standard implementation and it improves network lifetime by a factor of up to 6.5 [41].

C. Security Service Proxy (SSP) in RESTful Environments

In [42], Abeele F. *et al*. proposed reverse proxy approach to overcome end to end security and performance issues in constrained RESTful environments that were identified in many studies including their study [42] as well as other studies such as [43], [44]. The authors argue that the identified issues can be overcame using reverse proxy approach which splits the end-to-end security at the proxy. Accordingly, Secure Service Proxy (SSP) has been proposed which offers additional functionality and services on behalf of the constrained networks and nodes. The main goal of SSP is to reduce the load and improve the constrained RESTful environments performance and functionalities.

The proposed SSP is extending the constrained devices with wide range of features. It uses virtual devices which is associated with one or more endpoints and each endpoint is linked to the application or transport layer. Each virtual device has a unique IP address to allow the proxy to listen to its associated endpoints' traffic. Fig. 13 illustrates the SSP design [42].

Even though the communication will not be an end to end communication, the SSP will provide both parties with DTLS security contexts as well as translating all CoAP messages. Overall, the proposed SSP have benefits such as overcoming end to end security issues in constrained RESTful environments and allowing reverse proxy by implementing a virtual device for every constrained device [42].

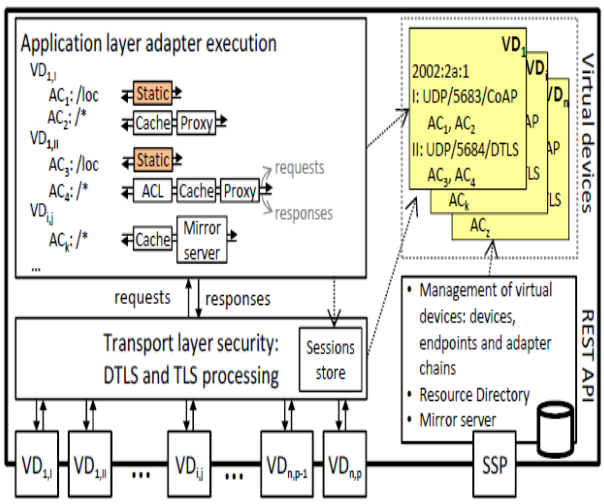

Fig. 12. SSP Design [42].

The authors evaluated the proposed SSP using two scenarios: In the first scenario, they allocated the proxy near to the constrained devices through assigning

addresses from a neighboring LAN network to the virtual devices. The second scenario is residing the proxy in an upper level of the constrained devices such as on the cloud. The experimental results in both scenarios showed that using SSP reduces the load significantly by reducing the processing time, network traffic, power consumption, network delay and packet loss rates for constrained devices. Thus, SSP helps to guarantee the proper operation of constrained networks and nodes. Although, SSP showed impressive results, it has some limitation. SSP is introducing Single point of failure in terms of security and operation. Also, SSP is vulnerable to lose all session, public and private key in case of compromising [42].

In [45], authors proposed a communication framework called Atlas for enabling interoperability among IoT devices speaking different languages by offering lightweight IoT protocol translator connecting heterogeneous devices via well-defined interfaces. The proposed framework introduces a lightweight communication on CoAP, REST protocols over HTTP, and MQTT.

G. Tanganelli *et al*. in [46] presented an edge-centric distributed architecture based on both the CoRE Resource Directory interface and (CoAP) to provide a discovery and access services in IoT and Fog Computing environment. In this study, the CoAP protocol has been implemented at gateway as a reverse-proxy such that all client CoAP requests are sent to the intermediary gateway rather than directly to IoT devices.

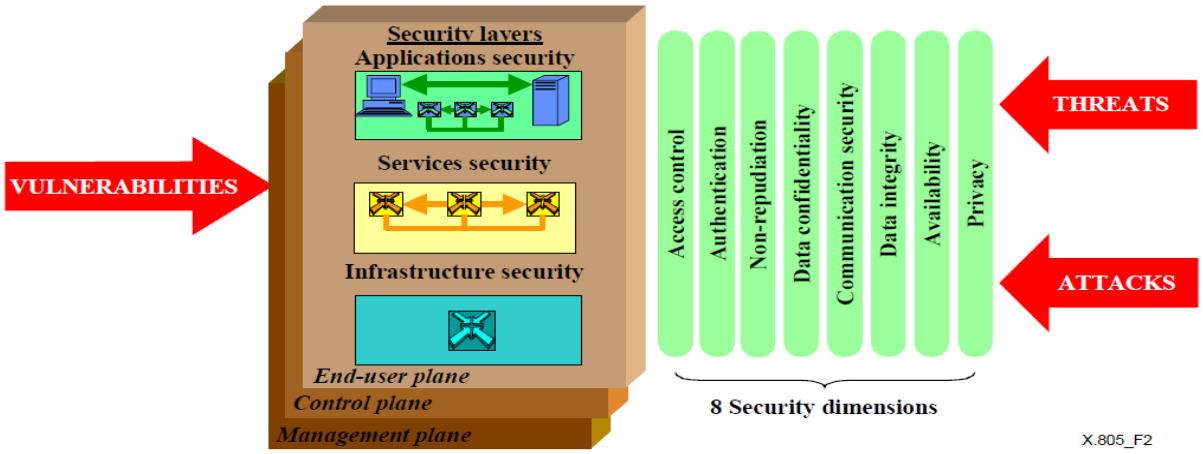

Fig. 13. X.805 security standard architecture [47].

D. DTLS and IPSec Implementations Using X.805 Security Standard

In [47], the authors evaluated DTLS and IPsec protocols which is proposed to be used to secure CoAP. Both DTLS and IPsec implementation have been analysed using X.805 security standard. X.805 architecture is a top-down systematic approach which is utilized to detect, predict, and correct the security vulnerabilities. It can be applied to any network element, service, and applications in order to investigate its vulnerabilities. Generally, it is defining three security layers, three security planes and eight security dimensions which are identified to address the general system vulnerabilities based on the network activities. Fig. 14 mentions all layers, planes and dimensions in details [47].

The experiments showed that using DTLS to secure CoAP is failed to meet some security requirements. First, even though the multicast communication is an essential feature of CoAP, it is not supported by DTLS. Second, DTLS handshake protocol can cause exhaustion attack which make the nodes lose their roles in the network which will lead to a complete disruption to the entire communication. Thirdly, DTLS countermeasure against replay attack is not guaranteed in all scenarios; thus, it is vulnerable to replay attack which will consume the resources. Lastly, DTLS handshake protocol is not provide end to end authentication [47].

In the other hand, IPSec provides different security services such as integrity, access control, data authentication, confidentiality, anti-replay mechanism, and limited traffic flow confidentiality. Encapsulating Security Payload Protocol (IPSec-ESP) is used to secure CoAP using IPSec. However, beside the drawbacks and problems of IPSec respect to Network Address Translation (NAT) and Port Address Translation (PAT), the encryption process of the small packets generates a large overhead to the network. Furthermore, kernel level modifications are needed to apply any modifications to IPSec because it is embedded in the IP stack. Additionally, configuring, managing or troubleshooting IPSec and Internet Key Exchange (IKE) are complex tasks and as the number of the constrained devices is increase the complexity will increase accordingly. Also, supporting multicast communication is difficult using IPSec [47]. Since IPSec and DTLS were not designed for constrained devices and environment, the investigating of both protocols implementations in the CoAP highlights

that both protocols failed to meet some security requirements as illustrated in Fig. 15.

Security Dimension	IPSec	DTLS
Access Control	No	No
Authentication	Yes	Partially- Server Only
Non-Repudiation	Yes/No; depends on authentication method. PKI not supported by constrained devices	Yes/No; depends on authentication method. PKI not supported by constrained devices
Confidentiality	Yes	Yes
Communication Security	Yes	Yes
Integrity	Yes	Yes
Availability	Mitigation- No full protection	Yes-stateless cookie
Privacy	No	No

Fig. 14. DTLS and IPSec analyses using X.805 [47].

E. Optimized DTLS and CoAP for IoT

In [48], Maleh Y. developed an optimized version of DTLS in terms of performance. It reduces the cost of DTLS communication and improve its weaknesses such as cookie exchange in the handshake process by integrating the DTLS inside CoAP. The integration of different encryption elements within the CoAP message added more security layer in addition to save energy and time when establishing the connection between the client and the server. To mitigate DoS attack, DTLS handshake process is extended with a cookie exchange technique. Toward avoid consuming the resources by DoS attacks, the client must demonstrate its capabilities to the server before it allocates the needed resources to the client. The experiment results showed that the proposed enhancement lead to better performance due to simplification of handshake process. Generally, the packet overhead and the energy consumption have been reduced as well as the ROM usage which is reduced by almost 23% [48].

In [49], they introduced a mechanism that optimized the latency of CoAP transportation in RTIoT based on forward error correction (FEC). The approach showed a compromising between loss and latency and according to their experiments the proposed model considerably lowers the application layer loss and hence increased the throughput of the communication in the domain of IoT.

Other studies have proposed enhancements to DTLS by working around its delegation mechanism [50]-[53]. For example, researchers in [50] proposed an enhancement to the DTLS when being used with CoAP protocol by separating the DTLS handshake phase and data encryption phase to solve the delay, overhead, loss problems caused by handshake packets and communication processes. The core idea in their approach is to use Secure Service Manager (SSM) to delegate the handshake phase and thus eliminating the space and power required by the constrained devices to do such handshake. To ensure the end-to-end security, the end node is responsible of doing the encryption and decryption of the data.

F. Security Aspect of CoAP-HTTP Proxy

In [54], the authors have been studied the security aspect of CoAP-HTTP proxy in details. Since CoAP allows cross protocol proxies between CoAP and HTTP, the security of CoAP has been analysed by performing vulnerability analyses of CoAP.

In order to analyse CoAP security, a lab has been configured using CoAP Python with Eclipse IDE, Californium (Cf), CoAP framework and Copper (Cu) Firefox add-on for security. First, Cooja simulator is used to create the virtual testing environment for CoAP. Similarly, Python and CoAPthon library have been used to develop the CoAP client, CoAP server with caching and observer, HTTP to CoAP forward proxy feature. Likewise, CoAP services are developed using Californium (Cf) which is an open source CoAP. The Copper (Cu) which is a user agent has been installed in the Firefox browser and it is interacting with other IoT services using 'CoAP' URI [54].

Once the environment has been configured and simulated, the CoAP security has been tested by running the CoAP client (Cu) in Firefox and Cf proxy in Eclipse IDE. After that, the CoAP client (cu) will access the CoAP server over the cf-proxy [CoAP to HTTP] conversion. Once the cu started the communication with Cf proxy, the communication will be intercepted using Burp Suite. Both Firefox and Burp Suite will be configured with port 8081. Once the packet is captured, its information can be analysed [54].

The results showed that, the data is transmitted in a plaintext format which is vulnerable for attacks. This proves that CoAP proxy is vulnerable to different types of attacks and data manipulation [54].

G. Lightweight Security Scheme for IoT Applications Using CoAP

Arijit U. et al. in [55], proposed a lightweight security scheme in CoAP for IoT application. The scheme is based on Advanced Encryption Standard (AES) algorithm with key length equals to 128. The scheme has two components: CoAPs-Lite which enables the lightweight security in CoAP, and Auth-Lite component which enables the lightweight authentication process in IoT devices.

The proposed scheme provides authentication and confidentiality. The authentication is carried out with symmetric key-based authentication in integration with key management. The shared key is exchanged using AES Cipher Block Chaining (CBC) mode. To avoid the handshake overhead, it is embedded in the payload and consist of two roundtrips only which consist secret distribution, session initiation, server challenge and sensor response phases. Also, to secure the authentication scheme against different attacks, the challenge-response process (nonce) technique has been applied. It is

generated using Random Number Generator appended with a counter[55].

In order to adopt the security scheme into CoAP, a few modifications to the header has been proposed to improve the security operations and minimize the overall communication overhead. A new option in CoAP header is introduced which is called "AUTH" to enable the secure mode which uses option to indicate the critical option class. Moreover, "AUTH_MSG_TYPE" option is introduced to provide variety in terms of the messages that can be used for establishing an authentication session [55].

To embed the authentication within CoAP; first the sensor-gateway will send a POST message with confirmable mode, "AUTH" option, "AUTH_MSG_TYPE" and device ID parameters in the payload. Then, the server will drive the device ID from payload and determine its pre-shared secret associated with this ID. Once the "AUTH", "AUTH_MSG_TYPE" options are received, the nonce and the session key will be generated, and the server will generate the encrypted payload, and the response will be sent back to sensor-gateway. Once the sensor-gateway received the response, it will decrypt it using the shared secret key. Then it will generate another nonce to send it back to the server and it will generate the encrypted payload using session key "K". Encrypted payload will be sent to the server with POST message. Both POST messages sent be the server and the sensor gate to authenticate must have the same token value. Once the server received the payload, it will decrypt it using "K" and checks the received nonce_1. If it is correct, the client will be authenticated and a response message with code "client authenticated" will be sent. Otherwise, the server will send "client not authenticated" message [55].

Correspondingly, once the client is authenticated, the confidentiality process will be embedded to all communications. The data will be encrypted and decrypted using the key "k", encrypted data will be posted using POST with option type "DEC_CONF" in the header. The server will send the status of the decryption process using response message to the client based on the value of "DEC_CONF" option. If it is true, the status will be sent; otherwise, not. According to the status of the decryption process, the client will decide to re-send the message or not. The experiment showed that the average decryption time in the server is 0.67 seconds which is much less that the re-transmission timeout of CoAP [55].

Since there are no public key crypto component, the proposed scheme considered as a faster than the public key-based systems. In general, the proposed system is efficient, generic, applicable to different IoT applications and resilient against replay and Man in the Middle attacks. Also, it has low overhead due to embedding the symmetric key-based authentication with integrated key management [55].

There are several studies to leverage the lightweight security scheme in CoAP such as in [56] where they proposed a lightweight security schema on CoAP that relies on enhancing the mechanism of authentication and key management for vehicle tracking systems. The main idea of their work is to utilize some unused CoAP header options to embed the authentication mode as well as to create a secure channel with a low communication cost for messages exchange in vehicle tracking systems.

H. Lightweight Security Protocol in Wireless Sensor Networks

In [57], Roh J. H. et al. designed lightweight security protocol which is utilizing the binary CDMA security codes in sensors networks. It is suitable for low power sensors and it is designed mainly for wireless network sensors where the transmitted data must be confidential. The authors discussed binary CDMA and its security requirements and how to apply it in sensors networks to prevent attacks. The designed protocol is efficient cryptographic protocol which provides block cipher algorithm, CDMA security, authentication, timestamp, monitoring and detection countermeasures. All these countermeasures have been added to the designed protocol to prevent possible attacks such as replay, sybil, sinkhole attacks and hello flooding.

The architecture of the sensor network consists of three node types in hierarchical structure which are Member node (MN), Cluster header (CH) and Base station (BS). MNs are classified into different clusters which is headed by CH. One of the main responsibilities of CH is time synchronization. As illustrated in Fig. 16, BS acts as a gateway between two or more clusters [57].

The designed protocol has been tested using 128 bits key size, different hash and block cipher algorithms have been tested to choose the most suitable algorithms. After testing MD5, SHA-1, SHA-256, SHA-512, RC5, IDEA, DES, 3DES/EDE and AES/CTR algorithms, MD5 and AES algorithms have been used for message authentication and encryption respectively. Both algorithms showed the lowest cycles per bytes with high performance comparing to other algorithms [57].

With the same goal, authors in [58] proposed a lightweight implementation of Internet Key Exchange protocol (IKEv2) to provide key agreement and authentication for IoT in the domain of Wireless Sensor Networks where it optimizes the communication overhead and the consumption of the energy at sensors. The main idea of the proposed technique is to calculate the hash value of both IP address and the ID instead of a digital certificate, hence, having a variant of IKEv2 that is a certificate-free implementation. The evaluation has been conducted on NS-2 and the results showed that the proposed algorithm outperforms the original version of IKEv2 in terms of energy consumptions and overall communication cost.

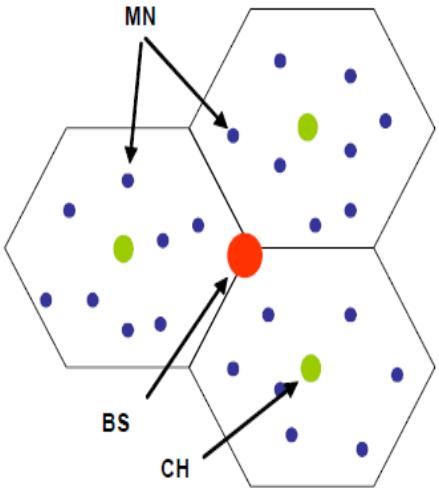

Fig. 15. Sensor network architecture [57].

I. Suite of Lightweight Security Protocols for IoT

Wu X. *et al*. in [59], presented a suite of lightweight security protocols which provides encryption, key management, identity authentication, and data integrity services. For encryption and instead of using the pre-shared cryptographic keys, the probabilistic encryption procedure or hashed keys are used in the encryption and decryption procedures to avoid compromising encrypted data in case if the keys are compromised. Although the encryption protocol is used, the key-for-one-file encryption is applied to ensure security which is illustrated by using One Time Pad (OTP). A random key will be used for every file or data item will to apply key-for-one-file encryption. Applying key-for-one-file encryption eliminates the ciphertext only attack, while applying probabilistic or hashed key is eliminating the chosen plaintext attack [59].

The key is chosen randomly by legitimate devices only from the key store which is large pool of random keys. The key is generated using key store seed which is pre-shared between the legitimate devices and stored in the device Hardware Security Module (HSM). Thus, when the attacker compromises the device and tries to extract the seed without passing the authentication, the device will automatically delete the seed to maintain the secrecy of keys. No two keys will be used repetitively, and all keys are uniquely determined by the key store seed. Also, the distribution of the key index doesn't disclose any information about the key itself which make it more secure [59].

The data integrity and originality are ensured using message authentication code (MAC) which is generated using the proposed cipher-based MAC algorithm. The identity authentication will be done during the configuration system for the new legitimate device when it requests joining the network for the first time. It will share its identity with the hub and other devices in the same network. First, device1 (the new device) will send w1 which is random selected key to device2 (in the same network) associated with encrypted time stamp and its ID. After that, device2 will generate k(w1) and verifies device1 by decrypting the received ID and time stamp in addition to compare it with the ID obtained from the hub. Then, device2 will randomly choose a new key w2 to send to device1 along with its encrypted ID and timestamp. Lastly, device1 will decrypt the ID of device2 and generate k(w2). Once the both devices are authenticated to each other, the communication can be started successfully. In order to ensure transmitted data integrity, the Cipher-based Message Authentication Code (CMAC) is used [59].

The proposed protocols performance has been tested and compared against to IPsec which needs computationally intensive resources. IPsec uses AES 128, SHA-1, DH, and RSA to ensure security; however, SHA-1 broken recently, while the security aspects of Diffie-Hellman and RSA algorithms mainly rely on the complexity and hardness of both discrete logarithm and integer factorization problems. In general, the proposed protocol achieves a higher level of security with a very low resources consumption. Fig. 17 presents a comparison between IPSec and proposed suit protocols [59].

In [60], they proposed a secure key management and user authentication scheme for Fog Computing environment by utilizing lightweight operations such as cryptographic hash function and bitwise exclusive-OR. The proposed approach has been implemented and evaluated using NS2 simulator and results showed its efficiency.

Security functions	IPsec	Proposed protocols
Encryption	AES-128 in CBC or Counter mode	Lightweight cipher with one-key-for-one-file encryption
Integrity Protection	HMAC SHA, or AES-128 cipher based MAC	Lightweight cipher based MAC
Key Management and peer authentication	Diffie-Hellman (for key exchange); RSA with SHA (for peer authentication)	Efficient and information theoretically secure key management; Lightweight cipher based authentication

Fig. 17. Comparison between IPSec and proposed suit protocols [59].

In [61], a key exchange protocol based on attribute-based encryption has been proposed to secure the communication between fog nodes. In this approach, they involved digital signature techniques with attribute-based encryption to achieve an efficient and effective secure communication system among fog nodes. The proposed

protocol has been evaluated and analyzed in terms of communication overhead, message sizes, and security aspects and results showed that it outperforms the certificate-based protocol.

J. Network Coding Technique for a Communication of Fog nodes

In [62], Marques B. et al. applied network coding technique to the Dissemination of Small Values data communication protocol between fog nodes of wireless sensors. The network coding technique can adopt different layers and network classes to improve the data flow. This protocol is a communication protocol used by network sensors to communicate. When there is an information on a sensor x which needs to be sent to neighboring nodes, the protocol is used to send the information in periods. It starts with shorter periods then the long periods and the same process will be carried out for all neighboring nodes.

In order to simulate the proposed protocol and evaluate its performance, the environment was simulated using TOSSIN simulator along with two applications which are any Dissemination of Small Values protocol application based and TrickleTimer timer. All the nodes have the same hardware and software characteristics and each sensor has unique ID starting from zero to the last topology ID. Two different topologies have been used which is Square and Tree. Both topologies have different number of nodes to evaluate its performance. Dissemination of Small Values protocol has been tested using two applications with and without Network Coding technique to compare the communication performance [62].

The proposed protocol obtained excellent results by optimizing the data flow, reducing the latency, and the data transmission. Also, the used bandwidth has been reduced when applying the network coding on the Dissemination of Small Values protocol. In general, applying Network Coding increases the protocol performance by 50%-60% of the original protocol [62].

K. Elliptic Curve Cryptography (ECC) Based Proxy for Fog Computing and IoT

Diro A. A. et al. [63] analyzed Fog Computing security challenges in general as well as proposing a novel encryption scheme which is Elliptic Curve Cryptography (ECC) based proxy re-encryption as a lightweight encryption scheme for communications between Fog Computing and IoT devices. It consists of five procedures which are: key generation, client encryption, fog encryption, fog decryption, and client decryption [63].

Among all distributed fog nodes, one fog node should be selected as a trusted authority or coordinator for the key generation procedure to produce the public curve parameters, public and private keys of the coordinator. Then the public elliptic curve parameters are distributed to all fog nodes and IoT devices. The trusted authority (chosen node) will send the KCi1 to IoT devices as a private key and Kci2 to other fog nodes, where ci is the identity of IoT device. After that, the IoT device will encrypt the message using its private key (KIDi1). Then, the corresponding fog node will re-encrypt the IoT device ciphertext using the fog node key (KCi2). The fourth procedure is decrypting the ciphertext by the corresponding fog node and convert it to an intermediate ciphertext that can by fully decrypted only by the corresponding IoT device. Lastly, IoT device Ci will fully decrypt the message using its private key KCi1 [63].

To evaluate the performance, the proposed scheme has been implemented using Java on top of nics-crypto to support proxy re-encryption. The execution time, throughput and ciphertext expansion are used to compare the performance of the proposed scheme and the RSA. Based on the experiment, it can be observed that using proxy ECC is faster than RSA in encryption and decryption processes of messages with different sizes. The RSA decryption process is considered as the slowest process while proxy ECC is much faster in decryption computations. Moreover, proxy ECC produces smaller ciphertext than RSA and its encryption throughput is higher than the corresponding RSA over various data seizes. As the message size increase, the encryption throughput of proxy ECC increases while RSA throughput is decreased. In general, the experiment has been proved the effectiveness and efficiency of the proxy ECC scheme as a cryptographic mechanism for the fog to things communication security 63].

In [64], authors proposed a novel lightweight security solution for publish-subscribe Fog Computing based on ECC protocol for key exchange and encryption. They analyzed the application of such lightweight schema in terms of scalability, overhead, and security metrics in Fog Computing environment. The study showed the applicability of using such protocol to secure the communication of Fog Computing due to the ECC properties: short key lengths, small message sizes, and low computation power requirements.

IV. Discussion and Analysis

Table I summarizes all proposed techniques to secure the communication between fog and IoT devices. The comparison has been carried out by comparing the used technique, study objectives, results and limitations for each paper. If "N/A" term is used it indicates that the limitation column is not applicable for the study.

As illustrated in the table and even though there were hard work to enhance CoAP security by optimizing DTLS implementations, introducing SSP, or compressing DTLS header in [41], [48], [42] respectively, these enhancements have limitations such as degrading the overall performance or testing it within simulated environments which differs than the real one.

Initially, DTLS was not designed specifically to be worked over constrained devices and authors in [65] demonstrated and proved by their conducted experiments

that DTLS and TLS implementations are feasible to different attacks in real environments. Moreover, authors in [47] investigated DTLS security based on X.805 security standard, and the results showed that DTLS has been failed to meet some important security aspects. Furthermore, DTLS does not suit the CoAP proxy and it is handshake process need six messages which overhead the communication channel and increases the number of transmitted bytes which consumes the energy. Energy parameter has significant impact for any constrained device because it has limited resources. Same issues have been faced with IPSec and HTTP protocols which is proposed to enhance CoAP security in [47], [54]. Thus, introducing new suit security protocols are much better to consider the constrained devices characteristics and parameters when designing these protocols.

TABLE I: SUMMARY OF RELATED WORK

Paper	Technique	Study Goals	Results	Limitations
[38]	LSPWSN	Proposing lightweight protocol to secure communication for Web Services in wireless sensors	In general, LSPWSN protocol presents a more stable behavior than CoAP.	Performance of LSPWSN is lower than CoAP due to TCP which is used without any compression mechanism
[41]	An optimized implementation of DTLS for CoAP, by integrating the DTLS protocol inside the CoAP.	Optimized implementation of DTLS for CoAP.	The experiments results show that the optimized ECC solution outperform the standard implementation and it improve network lifetime by a factor of up to 6.5.	Not tested in real environments.
[42]	Security Service Proxy (SSP).	To offer additional functionality and services on behalf of the constrained networks and nodes as well as improving the constrained environments performance.	Applying SSP reduces a significant processing, network traffic, power consumption, network delay and packet loss rates for constrained devices. Also, it guarantees the proper operation of constrained networks and nodes.	Introducing single failure point as well as its vulnerability to replay attacks.
[47]	Evaluating DTLS and IPSec implementations using X.805 security standard.	Investigating DTLS and IPsec implementations and effectiveness in securing CoAP.	The experiment results highlight that both protocols failed to meet some security requirements.	N/A.
[50]	Lightweight secure communication for CoAP-enabled Internet of Things using delegated DTLS handshake	Enhancing the DTLS by separating the DTLS handshake phase and data encryption phase by utilizing Secure Service Manager (SSM) as a mean for delegation.	Reduces the delay, overhead, loss problems caused by handshake packets and communication processes.	N/A.
[54]	Configuring a lap to study the security aspect of CoAP-HTTP proxy.	Analysing CoAP-HTTP proxy security.	The experiment proves that the data is transmitted in a plaintext and the CoAP proxy is vulnerable to different types of attacks.	N/A.
[55]	A lightweight security scheme in CoAP for IoT application. The scheme uses AES algorithm Cipher Block Chaining (CBC) mode with 128 key length.	Studying lightweight security scheme for IoT applications using CoAP.	It is efficient, generic and resilient to number of security attacks like replay and meet-in-the-middle attacks.	The pre-shared key must be hardcoded in the device otherwise this scheme will not work.
[48]	An optimized version of DTLS in terms of performance. Improving DTLS: cookie exchange in the handshake process by integrating the DTLS inside CoAP.	Enhancing and optimizing the performance of the DTLS network for CoAP.	The proposed enhancement lead to better performance in terms of packet overhead, handshake time processing, and energy consumption	Not tested in real environments.
[57]	A lightweight security protocol which is utilizing the binary CDMA security codes in sensors networks.	Designing a lightweight security protocol for sensors networks.	MD5 and AES algorithms have been used for message authentication and encryption respectively. Both algorithms showed the lowest cycles per bytes with high performance.	During the authentication and authorization phase. The neighboring nodes of the BS will suffer from high communication overhead due to forwarding request and response packets.
[58]	Lightweight key agreement protocol for IoT based on	Providing lightweight key agreement and authentication based on a hashing technique of IP address and ID instead	The evaluation has been conducted on NS-2 and the results showed that the proposed algorithm outperforms the original version	To prove the efficiency of the proposed algorithm, its required to conduct a

	IKEv2	of a digital certificate to optimize communication and computation costs.	of IKEv2 in terms of energy consumptions and overall communication cost.	performance evaluation against other TLS-based frameworks.
[59]	A suite of lightweight security protocols which provides encryption, key management, identity authentication, and data integrity.	Providing identity authentication, encryption, key management, and data integrity.	It achieves a higher level of security with a very low resources consumption	The authentication procedure in the proposed suit vulnerable to malicious insider attacks.
[62]	Appling a network coding technique to the Dissemination of Small Values data communication protocol between fog nodes composed of wireless sensors	Increasing the protocol performance	Applying Network Coding increases the protocol performance by 50%-60% of the original protocol.	The timer overloads the communication channel with resending data for long periods.
[63]	A novel encryption scheme which is Elliptic Curve Cryptography (ECC) based proxy re-encryption for fog to things as a lightweight encryption scheme.	Securing the communication by fast and robust cryptographic algorithm.	The proposed technique proved its effectiveness and efficiency of the proxy ECC scheme as a cryptographic mechanism for the fog to things communication security.	Not tested in real environments.

Different researchers proposed different lightweight protocols to suit the constrained devices. In [38], the LSPWSN protocol is efficient and secure which is proposed to be used for the Web Services in wireless sensors. Generally, the memory consumption and bandwidth achieved better than CoAP. However, it degrades the overall performance and it is proposed only to be used for wireless sensor. Since sensors must send and receive data in real time, LSPWSN might be not applicable due to performance degradation. Further improvements are requested to enhance LSPWSN performance.

In [55], the proposed a lightweight is based on AES algorithm with128 bits key length. It has excellent features such as speed, efficiency, applicability to different IoT applications, resilience against replay and Man in the Middle attacks, as well as its low overhead. However, authors assumed that the pre-shared key is hardcoded into each device at the time of manufacturing, which is not always applicable. Improving the scheme to make it work without this assumption is recommended to evaluate the proposed technique performance.

In [57], the designed lightweight security protocol is utilizing the binary CDMA security codes in sensors networks. It is designed mainly for wireless network sensors where the transmitted data must be confidential. This proposed technique has problem with the BS, it will be overloaded with packets when neighboring nodes of BS are forwarding request and response packets.

In [59], the lightweight security suite is proposed to provide encryption, key management, identity authentication, and data integrity services. The overall performance achieves a higher level of security with a very low resources consumption. However, there are a possibility of malicious insider attack in the authentication procedure. This possibility is low and unlikely to be happened; however, it is recommended to be reduced furthermore. In [62], authors applied network coding technique to the Dissemination of Small Values data communication protocol between fog nodes of wireless sensors. It is simulated to measure its performance; it is obtained excellent results by optimizing the bandwidth usage, data flow, reducing the latency, and the data transmission. However; sometime the used timer is overloading the communication channel. A novel encryption scheme based on ECC is proposed to encrypt the data. It produces smaller ciphertext in less time compared to RSA; however it's not tested in real environments and it's known that ECC is complex and hard to implement [63].

V. Conclusion

Fog Computing Concept has been introduced to address the cloud computing limitations and to provide multiple benefits such as low latency and mobility. However, the security of Fog Computing protocols is still a concern especially those related CoAP protocol. CoAP protocol does not have the reliable standards to secure its architecture. The biggest challenge faced by CoAP protocol is the lack of the main researches on how security can be managed or provided to the protocol. Although there were different studies to secure CoAP using DTLS and optimizes its implementations, it is proven that DTLS is not applicable and it was not designed for IoT devices. Thus, different researchers worked on proposing new lightweight protocols which are suitable for limited resources devices.

This paper surveyed the main security mechanisms proposed to secure CoAP protocol, its architecture, security and performance evaluation. There are different techniques which must be improved furthermore and tested to evaluate their performance. In conclusion, all related studies and techniques have been surveyed and discussed in deep to provide a solid reference for the Fog Computing and CoAP protocol security. As a

contribution, all reviewed papers have been categorized into two categories and a taxonomy has been proposed to ease understanding all available techniques to secure CoAP. It is found that using DTLS to secure CoAP is not applicable at all and it is not designed for constrained devices. Even though there were excellent studies to optimize the DTLS implementations in CoAP, it is still not applicable. Thus, different researchers proposed different lightweight protocols to provide constrained devices with security services such as encryption, authentication, authorization, and integrity.

LSPWSN performance in [38] must be improved furthermore to be at least equal to CoAP performance. In [55], the authors assumed that the pre-shared key is hardcoded into each device at the time of manufacturing, which is hard and not applicable at all. It is recommended to propose similar scheme without this assumption and evaluates its performance and security. It is recommended in [57] to solve BS problem by finding a technique to avoid packet overloading when sending too much packets between neighboring nodes. Also, It is recommended to test [58], [63] techniques in realistic environment to evaluate its performance and improve it furthermore if needed. Proposed technique in [63] showed impressive encryption results; however it is utilizing ECC which may be difficult to be implemented on IoT device. Among all reviewed studies, [58], [59], [62] showed robustness performance with excellent security services. It is still need minor improvements such as testing in real environments and mitigating minor issues. Since DTLS is not designed for constrained devices, future researchers must focus on proposing new lightweight protocols or improving existing ones. IoT-Fog communications must be secured, and different security services must be considered while designing these protocols. Also, the IoT devices characteristics such as resources limitations must be considered. As a future work, improvements and recommendations mentioned previously must be taken in consideration as well as evaluating other techniques [66]-[68] in securing constrained devices.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Ebtesam conducted the survey under the supervision of Fahd, both authors discussed and analyzed the survey work and wrote the paper. Fahd reviewed the entire paper and all authors had approved the final version

REFERENCES

[1] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-Art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416-464, 2018.

[2] K. Lee, D. Kim, D. Ha, U. Rajput, and H. Oh, "On security and privacy issues of fog computing supported internet of things environment," *J. Neurosci. Res.*, vol. 58, no. 3, 2015.

[3] L. M. Vaquero and L. Rodero-merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[4] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained application protocol (CoAP)," *Internet Engineering Task Force (IETF): Fremont*, CA, USA, 2014.

[5] Z. Shelby, K. Hartke, and C. Bormann. (2014). The constrained application protocol (CoAP). [Online]. Available: https://tools.ietf.org/html/rfc7252 (

[6] A. Banks and R. Gupta. (2014). OASIS Message Queuing Telemetry Transport (MQTT). version 3.1.1, OASIS. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[7] AMQP - Advanced Message Queuing Protocol. [Online]. Available: https://www.amqp.org/

[8] DDS – Data Distribution Services. [Online]. Available: http://portals.omg.org/dds/

[9] Zigbee. [Online]. Available: http://www.zigbee.org/

[10] ISO, UPnP - ISO/IEC 29341-1:2011 Device Architecture. [Online]. Available: https://www.iso.org/standard/57195.html

[11] OASIS, Devices Profile for Web Services Version 1.1. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html

[12] M. Iglesias-Urkia, A. Orive, Adri, A. Urbieta, and D. Casado-Mansilla, "Analysis of CoAP implementations for industrial Internet of things: A survey," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 7, pp. 2505—2518, 2019.

[13] D. Thangavel, X. Ma, A. Valera, H. X. Tan, and C. K. Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *Proc. IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Singapore, 2014, pp. 1–6.

[14] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," in *Proc. IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux*, Namur, 2013, pp. 1-6.

[15] M. Motaharul, Z. Khan, and Y. Alsaawy, "A framework for harmonizing internet of things (IoT) in cloud: Analyses and implementation," *Wireless Networks*, pp. 1-12, 2019.

[16] S. Bandyopadhyay and A. Bhattacharyya, "Lightweight internet protocols for web enablement of sensors using constrained gateway devices," in *Proc. International Conference on Computing, Networking and Communications*, 2013, pp. 334–340.

[17] O. Bergmann, K. T. Hillmann, and S. Gerdes, "A CoAP-gateway for smart homes," in *Proc. International*

Conference on Computing, Networking and Communications, Maui, HI, 2012, pp. 446-450.

[18] S. M. Chun and J. T. Park, "Mobile CoAP for IoT mobility management," in *Proc. 12th Annual IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, 2015, pp. 283-289.

[19] A. Betzler, C. Gomez, I. Demirkol, and M. Kovatsch, "Congestion control for CoAP cloud services," in *Proc. Emerging Technology and Factory Automation*, Barcelona, 2014, pp. 1-6.

[20] H. A. Khattak, M. Ruta and E. Di Sciascio, "CoAP-based healthcare sensor networks: A survey," in *Proc. 11th International Bhurban Conference on Applied Sciences and Technology*, Islamabad, 2014, pp. 499-503.

[21] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22-32, 2014.

[22] C. P. Kruger and G. P. Hancke, "Implementing the internet of things vision in industrial wireless sensor networks," in *Proc. 12th IEEE International Conference on Industrial Informatics*, Porto Alegre, 2014, pp. 627-632.

[23] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," *Internet Engineering Task Force (IETF): Fremont*, CA, USA, 2012.

[24] J. Vishwesh and M. B. Rajashekar, "Internet of Things (IoT): Security analysis & security protocol CoAP," *International Journal of Recent Trends in Engineering & Research*, vol. 3, no. 3, pp. 417–425, 2017.

[25] M. Zolanvari, "IoT security: A survey," *Computer Scientists & Computer Engineers at WashU*, 2015.

[26] N. Sabu. (2015). Fog Computing technology (PPT). [Online]. Available: https://www.slideshare.net/NikhilSabu/fog-computing-technology?from_action=save

[27] S. Yi, C. Li, and Q. Li, "A Survey of fog computing: concepts, applications and issues," in *Proc. Workshop on Mobile Big Data*, New York, 2015, pp. 37–42.

[28] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. First edition of the MCC Workshop on Mobile Cloud Computing*, New York, 2017, pp. 266–277.

[29] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Hong Kong J. Paediatr.*, vol. 18, no. 2, pp. 105–116, 2013.

[30] A. Dasgupta and A. Q. Gill, "Fog computing challenges: A systematic review," in *Proc. Australasian Conference on Information Systems Dasgupta & Gill*, Hobart, Australia, 2017.

[31] A. Verma, A. Soni, and P. Patel, "Fog computing: security, issues and its challenges," *International Journal of Engineering Research in Computer Science and Engineering*, vol. 3, no. 10, pp. 60–65, 2016.

[32] J. Dizdarevic, F. Carpio, A. Jukan, and X. Masip-Bruin, "Survey of communication protocols for internet-of-things and related challenges of fog and cloud computing integration," *ACM Computing Surveys*, vol. 51, pp. 1–29, 2019.

[33] S. Zamfir, T. Balan, I. Iliescu, and F. Sandu, "A security analysis on standard IoT protocols," in *Proc. International Conference on Applied and Theoretical Electricity*, Craiova, 2016, pp. 1-6.

[34] H. Lin and N. W. Bergmann, "IoT privacy and security challenges for smart home environments," *Information*, vol. 7, 2016.

[35] D. B. Ansari, Atteeq-Ur-Rehman, and R. A. Mughal, "Internet of Things (IoT) protocols: A brief exploration of MQTT and CoAP," *International Journal of Computer Applications*, vol. 179, no. 27, pp. 9–14, 2018.

[36] M. P. Singh and A. K. Chopra, "The internet of things and multiagent systems: Decentralized intelligence in distributed computing," in *Proc. IEEE 37th International Conference on Distributed Computing Systems*, Atlanta, GA, 2017, pp. 1738-1747.

[37] X. Chen, "Constrained application protocol for internet of things," Research Paper on Computer Science & Engineering, Washington University, 2014.

[38] P. E. Figueroa, J. A. Pérez, and I. Amezcua, "Performance evaluation of lightweight and secure protocol for wireless sensor networks: A protocol to enable web services in IPv6 over low-power wireless personal area networks," *International Journal of Distributed Sensor Networks*, vol. 13, no. 6, 2017.

[39] H. Wang and Z. Sun, "Compression method for IPSec over 6LoWPAN," *KSII Transactions on Internet & Information Systems*, vol. 12, no. 4, pp. 1819-1831, 2018.

[40] R. Garg and S. Sharma, "Modified and Improved IPv6 Header Compression (MIHC) scheme for 6LoWPAN," *Wireless Personal Communications*, vol. 103, no. 3, pp. 2019-2033, 2018

[41] A. Capossele, V. Cervo, G. D. Cicco, and C. Petrioli, "Security as a CoAP resource: An optimized DTLS implementation for the IoT," in *Proc. IEEE International Conference on Communications*, London, 2015, pp. 549-554.

[42] F. Van den Abeele, I. Moerman, P. Demeester, and J. Hoebeke, "Secure service proxy: A CoAP(s) intermediary for a securer and smarter web of things," *Sensors (Switzerland)*, vol. 17, no. 7, 2017.

[43] M. Vučinić, B. Tourancheau, T. Watteyne, F. Rousseau, A. Duda, R. Guizzetti, and L. Damon, "DTLS Performance in Duty-Cycled Networks," in *Proc. International Symposium on Personal, Indoor and Mobile Radio Communications*, Hong Kong, 2015.

[44] H. Tschofenig and T. Fossati," Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) profiles for the internet of things," *Internet Engineering Task Force (IETF): Fremont*, CA, USA, 2016.

[45] A. E. Khaled and S. Helal, "Interoperable communication framework for bridging RESTful and topic-based communication in IoT," *Future Generation Computer Systems*, vol. 92, pp. 628-643, 2019.

[46] G. Tanganelli, C. Vallati, and E. Mingozzi, "Edge-Centric distributed discovery and access in the internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 425-438, 2018.

[47] T. A. Alghamdi, A. Lasebae, and M. Aiash, "Security analysis of the constrained application protocol in the internet of things," in *Proc. Second International Conference on Future Generation Communication Technologies*, London, 2013, pp. 163-168.
[48] Y. Maleh, A. Ezzati, and M. Belaissaoui, "An enhanced DTLS protocol for internet of things applications," in *Proc. International Conference on Wireless Networks and Mobile Communications*, Fez, 2016, pp. 168-173.
[49] R. Herrero and D. Hernandez, "Forward error correction in real-time internet of things CoAP-based wireless sensor networks," *IET Wireless Sensor Systems*, vol. 9, no. 1, pp. 42-47, 2019.
[50] J. Park and N. Kang, "Lightweight secure communication for CoAP-enabled internet of things using delegated DTLS handshake," in *Proc. International Conference on Information and Communication Technology Convergence*, Busan, 2014, pp. 28-33.
[51] R. Hummen, J. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, "Towards viable certificate-based authentication for the internet of things," in *Proc. 2nd ACM Workshop on Hot Topics on Wireless Networks Security and Privacy*, ACM, 2013, pp. 37-42.
[52] R. Hummen, H. Shafagh, and S. Raza, "Delegation-based authentication and authorization for the IP-based internet of things," in *Proc. 11th IEEE International Conference on Sensing, Communication, and Networking*, Singapore, 2014.
[53] J. Park, H. Kwon, and N. Kang, "IoT—Cloud collaboration to establish a secure connection for lightweight devices," *Wireless Networks*, vol. 23, no. 3, pp. 681–692, 2017.
[54] D. M. Rathod and S. Patil, "Security analysis of constrained application protocol (CoAP): IoT protocol," *International Journal of Advanced Studies in Computers, Science and Engineering*, vol. 6, no. 8, pp. 37-41, 2017.
[55] U. Arijit, B. Soma, B. Abhijan, P. Arpan, and B. Tulika, "Lightweight security scheme for IoT applications using CoAP," *International Journal of Pervasive Computing and Communications*, vol. 10, no. 4, pp. 372-392, 2014.
[56] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, and A. Pal, "Lightweight security scheme for vehicle tracking system using CoAP," in *Proc. International Workshop on Adaptive Security*, ACM, no. 3, 2013.
[57] J. H. Roh, M. Y. Kim, and H. K. Moon, "An approach to designing lightweight security protocol on binary CDMA sensor networks," in *Proc. International Conference on Ultra-Modern Telecomm & Workshops*, 2009, pp. 1–6.
[58] M. Lavanya and V. Natarajan, "Lightweight key agreement protocol for IoT based on IKEv2," *Computers & Electrical Engineering*, vol. 64, pp. 580-594, 2017.
[59] X. W. Wu, E. H. Yang, and J. Wang, "Lightweight security protocols for the internet of things," in *Proc. IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications*, 2017, pp. 1–7.
[60] M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos, "Design of secure key management and user authentication scheme for fog computing services," *Future Generation Computer Systems*, vol. 91, pp. 475-492, 2019.
[61] A. Alrawais, A. Alhothaily, C. Hu, X. Xing, and X. Cheng, "An attribute-based encryption scheme to secure fog communications," *IEEE Access*, vol. 5, no. 1, pp. 9131-9138, 2017.
[62] B. Marques, I. Machado, A. Sena, and M. C. Castro, "A communication protocol for fog computing based on network coding applied to wireless sensors," in *Proc. International Symposium on Computer Architecture and High-Performance Computing Workshops*, 2017, pp. 109–114.
[63] A. A. Diro, N. Chilamkurti, and Y. Nam, "Analysis of lightweight encryption scheme for fog-to-things communication," *IEEE Access*, vol. 6, pp. 26820–26830, 2018.
[64] A. A. Diro, N. Chilamkurti, and N. Kumar, "Lightweight cybersecurity schemes using elliptic curve cryptography in publish-subscribe fog computing," *Mobile Networks and Applications*, vol. 22, no. 5, pp. 848-858, 2017.
[65] N. J. Al Fardan and K. G. Paterson, "Lucky thirteen: breaking the TLS and DTLS record protocols," in *Proc. IEEE Symposium on Security and Privacy*, Berkeley, CA, 2013, pp. 526-540.
[66] N. U. Ain and A. Rahman, "Quantum cryptography: A comprehensive survey," *Journal of Information Assurance and Security*, vol. 11, no. 1, pp. 31-38, 2016.
[67] A. Rahman, S. A. Alrashed, and A. Abraham, "User behavior classification and prediction using FRBS and linear regression" *J. of Information Assurance and Security*, vol. 12, no. 3, pp. 86-93, 2017.
[68] A. Rahman, N. Saba, and N. Ain, "A novel robust watermarking technique using cubic product codes," *Journal of Information Security Research*, vol. 6, no. 1, pp. 14-24, 2015.

Fahd Al-Haidari is currently an assistant professor at College of Computer Science and Information Technology (CCSIT), Imam Abdulrahman Bin Faisal University (IAU), Dammam, Saudi Arabia. He received his M.S. degree in Computer Science from King Fahd University of Petroleum and Minerals (KFUPM) in May 2006, and the Ph.D. degree in Computer Science and Engineering from the same department in 2012. His research interests include cloud computing, network security, algorithms and simulation, machine learning, cryptography, in addition to performance evaluation of computer networks.

Ebtesam J. Alqahtani is currently an Information Security Analyst with Saudi Aramco. She has recently finished her MS in Computer Science from College of Computer Science and Information Technology (CCSIT), Imam Abdulrahman Bin Faisal University (IAU), Dammam, Saudi Arabia. Her research interests include information security, data mining, AI and Data/Network Security.