Computer Systems Science & Engineering
DOI:10.32604/csse.2022.020888
Article

Tech Science Press

Lightweight and Secure Mutual Authentication Scheme for IoT Devices Using CoAP Protocol

S. Gladson Oliver[1,*] and T. Purusothaman[2]

[1]Department of Information Technology, Government College of Technology, Coimbatore, 641013, India
[2]Department of Electronics and Communication Engineering, Government College of Technology, Coimbatore, 641013, India
*Corresponding Author: S. Gladson Oliver. Email: gladsonresearch@yahoo.com
Received: 11 June 2021; Accepted: 12 July 2021

Abstract: Internet of things enables every real world objects to be seamlessly integrated with traditional internet. Heterogeneous objects of real world are enhanced with capability to communicate, computing capabilities and standards to interoperate with existing network and these entities are resource constrained and vulnerable to various security attacks. Huge number of research works are being carried out to analyze various possible attacks and to propose standards for securing communication between devices in internet of things (IoT). In this article, a robust and lightweight authentication scheme for mutual authentication between client and server using constrained application protocol is proposed. Internet of things enables devices with different characteristics and capabilities to be integrated with internet. These heterogeneous devices should interoperate with each other to accumulate, process and transmit data for facilitating smart services. The growth of IoT applications leads to the rapid growth of IoT devices incorporated to the global network and network traffic over the traditional network. This scheme greatly reduces the authentication overhead between the devices by reducing the packet size of messages, number of messages transmitted and processing overhead on communicating devices. Efficiency of this authentication scheme against attacks such as DoS (denial of service), replay attacks and attacks to exhaust the resources are also examined. Message transmission time reduced upto 50% of using proposed techniques.

Keywords: IoT; CoAP; AES; encryption; message transmisssion

1 Introduction

IoT combines various technologies like embedded technology, wireless and wired communication networks, data analytics, cloud computing and various protocols and standards to provide seamless and secure integration. The ultimate goal of internet of things is to provide connectivity anytime, anywhere for anything without human intervention by making 'everything' smart. It is being implemented in many application areas such as healthcare, smart cities, manufacturing, logistics, transportation, e-governance, infrastructure and many more [1]. These applications ranges from simple temperature monitoring, object tracking and surveillance to more complicated applications like industry process automation and predicting

events using machine learning. It is forecasted that 4100 crores of IoT devices are to be utilized in 2022 which will cause 8.9 dollar trillion market as given by an international data corporation (IDC) report [2].

Internet of things enables devices with different characteristics and capabilities to be integrated with internet. These heterogeneous devices should interoperate with each other to accumulate, process and transmit data for facilitating smart services. The growth of IoT applications leads to the rapid growth of IoT devices incorporated to the global network and network traffic over the traditional network. The existing information and communication network is already facing many security challenges and the growth of internet of things adds even more security challenges to the network and devices due to the limitations and constraints of IoT devices such as limited power supply, open-environment deployment, mobility of devices, lack of end-point autonomy and number of end point devices [3]. Many of the smart IoT devices manufactured are not built with required security standards which leads to massive security breaches. IoT networks will be targeted by 25% of known enterprise attacks by the year 2020, according to Gartner and hence, IoT security requires universally accepted IoT standards which have not yet been proposed.

In this article contribution is as follows, a lightweight and efficient technique for authentication which uses constrained application protocol (CoAP) between client and server is proposed. In this technique, client's identity is verified by CoAP server and server's identity is verified by CoAP client and authentication process is initialized by client. Authentication of devices by verifying their identity before starting actual communication is very much needed. Otherwise, the participating devices can easily be attacked and compromised by malicious user and the whole network may be compromised.

Our authentication mechanism takes only one round trip message to complete the authentication between client and server which greatly reduces the number of messages transmitted between client and server and hence the network traffic is reduced. This will also improve the performance of IoT nodes because of the reduction of number of messages to be processed by nodes. In this proposed authentication scheme, constrained application protocol based messages are transmitted between client and server for authentication which can be used as an alternative for Datagram Transport Layer Security (DTLS). Our authentication scheme replaces DTLS because of the computational complexity and high resource consumption but also incorporates the features presented by DTLS protocol.

The paper is organized into following sections. The previous research works relevant to mutual authentication between IoT devices are presented in section II. In section III, we illustrate proposed lightweight authentication scheme for mutual authentication between CoAP server and CoAP client. In section IV, the proposed technique is analyzed against various known attacks. In section V, the performance of the proposed technique is discussed and the results obtained by evaluating the technique are compared with various other existing authentication schemes. Section VI provides future directions for this research work and concludes the work.

2 Related Work

Here, various research works related to security of IoT objects and protocols and standards used for securing IoT objects are discussed. We also discuss about various authentication techniques used between IoT devices and their advantages and disadvantages.

In traditional internet, the client request for resources using Hyper Text Transfer Protocol (HTTP). The resource in the server is represented to reflect the present, past or desired state and this representation is communicated using protocol and this implementation of HTTP is known as representational state transfer (REST) [4]. HTTP request messages and HTTP response messages exchange data between HTTP server and HTTP client. This protocol is inefficient and too heavy to be implemented on devices that are powered by battery and constrained [5]. Hence, HTTP does not suit for the IoT devices as most of them

are resource constrained. So there is a need for a protocol to be used in constrained devices those are lightweight and robust. By keeping this in mind, the IETF (Internet Engineering Task Force) formed a team to work on RESTful constrained environments. This team has developed a protocol named CoAP (Constrained Application Protocol). This protocol is an application layer protocol optimized to Web of Things smart objects which are mostly resource constrained and powered by batteries [6].

Colitti, et al. [7] have compared life of battery, power consumption and number of bytes transferred per transaction while using CoAP protocol and HTTP protocol. The results are given as below in Tab. 1.

Table 1: Comparison of HTTP and CoAP on resource consumption (Source: Colitti et al.)

Protocol	Bytes per transaction	Power (in mW)	Life time (in Days)
CoAP	154	0.744	151
HTTP	1451	1.333	84

Hence, we choose CoAP protocol for the communication between IoT client, which need to observe the resources and IoT server which sends data to the observer.

For securing communication using HTTP over TCP (Transmission Control Protocol), TLS (Transport Layer Security) is employed. CoAP is secured over UDP (User Datagram Protocol) using DTLS (Datagram Transport Layer Security). DTLS is same as TLS but with some added functionalities to handle unreliable communication of UDP [8]. The applicability of a DTLS mode for an application based on CoAP should be weighed by considering resources available in the constrained nodes, cipher suites being used, session maintenance and the increased network overhead. All DTLS modes are not applicable in some constrained nodes and networks with some specific cipher suites because of the added significant implementation complexity and some handshake overhead for setting up security association between communicating parties.

In [9], the proposed implementation of DTLS + CoAP is focusing only on communication between android smartphones. The algorithms used in combination with DTLS are not suitable for limited resource IoT nodes. Usage of DTLS for well constrained sensor devices is not suitable because of public key crypto-system computational overhead and lengthy handshaking process overhead [10]. Even though some security schemes with low overhead such as raw public key can be used, the minimum 25 bytes per-packet overhead of DTLS occupies the one-third of usable frame. DTLS also imposes severe limitation in terms of round-trip messages when retrofitted to CoAP [11]. The authors Raza et al. [12], have proposed the idea of header compression to make CoAP-security lightweight. But this header compression does not yield much gain in performance as the DTLS itself is computationally heavy. The authors Freeman et al. [13] have proposed a scheme, which allows a client to delegate certificate path construction and certification path validation to a server. But typical use of this protocol is expected to be over HTTP. It relies on a third party other than communication parties and also imposes high communication overhead. In [14], Kothmayr et al., have proposed an end-to-end architecture for IoT security and authors have used DTLS and other existing standards and protocols. RSA is used in combination with DTLS which incurs high computational complexity and network overhead. A fully authenticated handshake in DTLS, where server and client should mutually authenticate each other, takes more than three round-trip times.

Authors of [15] have proposed a lightweight authentication technique using symmetric key algorithm Advanced Encryption Standard (AES). In this authentication technique, message payload is used to challenge server and client mutually to ensure the genuineness of identity of devices. Four messages are

exchanged between server and client for the completion of authentication process. In [16], Park et al. have proposed a system architecture that uses delegation method for end to end communication. In this system, the handshake process is delegated to a Secure Service Manager (SSM). This architecture requires trust manager, sensors and secure service manager to be located within the constrained networks. So, it is not feasible for all types of IoT systems. In [17], Granjal et al. have proposed an end-to-end security scheme, using DTLS handshaking. They have used Elliptic Curve Cryptography (ECC) for encrypting data in constrained devices. Even though ECC consumes less power and memory than other public key encryption methods, it consumes more power and memory than symmetric key encryption techniques [18]. Usage of complex cipher suites lead to high consumption of energy. Authors proposed an authentication scheme for resource constrained devices. They have replaced DTLS based authentication by payload based authentication technique which uses AES symmetric key cryptography for encryption between CoAP server and CoAP client in four-way handshake. The efficiency of this authentication scheme is not evaluated using any experimental analysis. The four way handshake authentication using CoAP message payloads. They have added two header options to the CoAP message header for enabling the secure mode. In [19], authors have proposed an authentication technique using four-way handshake and have modified header fields of CoAP messages. AES algorithm with 128 bit key is used to encrypt messages between CoAP client and CoAP server. UDP protocol is used in transport layer without using DTLS for achieving security. We have attempted to cut down the number of messages exchanged between server and client during authentication process without compromising the efficiency.

DTLS protocol can be used in combination with many different cipher suites. But this protocol need to be designed to support the devices with sufficient power supply and processing capabilities. So customizing DTLS to suit constrained nodes does not yield better performance. Hence, alternate security schemes are being proposed by various researchers. We propose an authentication technique without DTLS and evaluate the performance of this authentication technique to show that it suits well for constrained devices of IoT.

3 CoAP Payload Based Lightweight Authentication

In many of the applications of IoT, CoAP is used as a protocol in the application layer. CoAP protocol does not implement any security for transmitting information between two nodes [20]. DTLS can be used for securing the communication between CoAP server and CoAP client. But, as discussed in previous section, DTLS imposes high processing complexity to the constrained nodes and network overhead. It also needs more than four handshake messages to complete mutual authentication between communicating parties.

In this section, we propose an authentication technique between CoAP server and CoAP client using CoAP message payload. It avoids the overhead of implementing additional protocol to secure the communication. We use only two handshake messages to complete the mutual authentication between devices. Total data bytes transmitted between client and server for authenticating each other is around 300 bytes only. Total authentication time includes one round-trip time and processing time spent on both the constrained nodes.

We assume an IoT network with multiple clients and one server and the clients can observe the resources in the server. Fig. 1 shows the network assumed to illustrate the scenario.

All the clients and server are using the CoAP protocol in the application layer for observing resources in the server. To observe a resource, client should prove its genuineness of identity to the server. Once client is authenticated by server, the server will create session for that client and responds to the client. Then, the client will ensure the genuineness of server after receiving reply message from server. Once the server and client are authenticated mutually, they can exchange application data between them using session key generated only for that session.

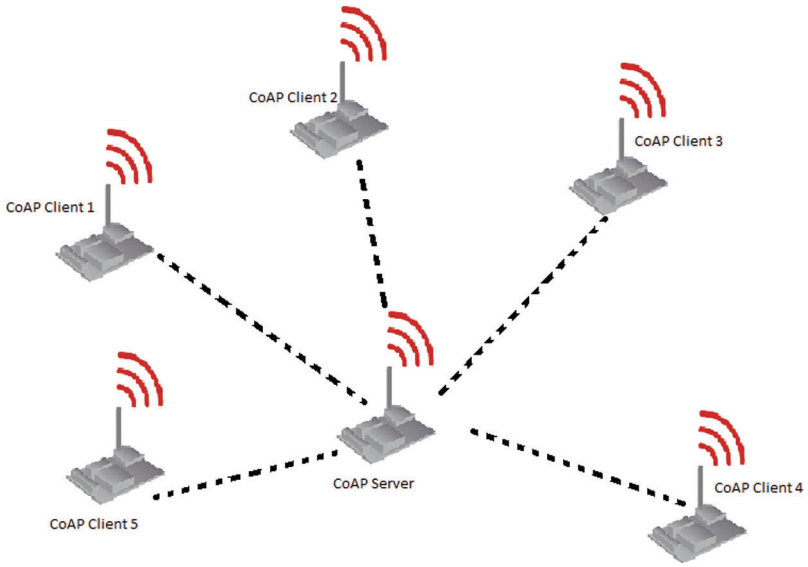

Figure 1: Assumed scenario

We use AES symmetric key algorithm for encrypting and decrypting data in CoAP client and CoAP server. It is a block cipher algorithm with 128 bits of constant block size. It has key sizes of 256 bits, 192 bits and 128 bits length. It is believed that the 128 bit key is sufficient for encryption and decryption in constrained devices of IoT system as they have low processing capabilities. AES encryption with 192 bit key and 256 bit key are having high computational complexity and demand advanced software and hardware platforms [21–23]. 256 bit key AES encryption takes 1.5 times more energy than 128 bit key AES encryption [18]. AES encryption technique is currently safe against brute force attack due to its large key size and the attacks proposed in literature are still not computationally feasible in AES [24].

In our technique, the client and server are pre-shared with 128 bits AES key. The keys are hardcoded into the devices during manufacturing phase and the end devices are assumed to be safe from physical attacks in accordance with internet threat model [25]. All devices are having a 64-bit unique device identifier assigned during manufacturing phase.

Our proposed authentication technique is having three steps namely one-time device configuration, client authentication and server authentication.

In the one-time device configuration phase, 128 bit AES key is hardcoded to every device and a 64-bit unique identifier is also assigned to every device. The server device is configured with a table which stores device identities (IDi) and shared secret keys (Ki) of every devices. This table is used to verify the device identity and find the AES key for that device and this is populated at the time of initial configuration of server device.

In client authentication phase, the identity of client device is verified by server and authenticated by server. The client generates a 64-bit nonce (Nc) and performs exclusive OR with the 64-bit device-ID (IDi). The 64-bit nonce is also exclusive- ORed with least significant 64-bits of the 128 bits shared secret of the client (KiL).

$$M1p = IDi \;||\; C1 \tag{1}$$

where, $C_1 = E\ \{K_i, ((N_c \oplus ID_i) \;||\; (Nc \oplus K_iL))\}$

Where M1p is the payload of CoAP message sent by client i and KiL is the low order 8 bytes of pre-shared key Ki of the client i. The total size of the message payload is 192 bits long including the unencrypted 64 bit long device identity. M1p is a CoAP confirmable message. Confirmable messages should be acknowledged by the response from the receiver of the message. CoAP also uses acknowledgement, non-confirmable and reset messages. The client sends the M1p to server. The Fig. 2 shows messages transmitted between client and server during the authentication process.

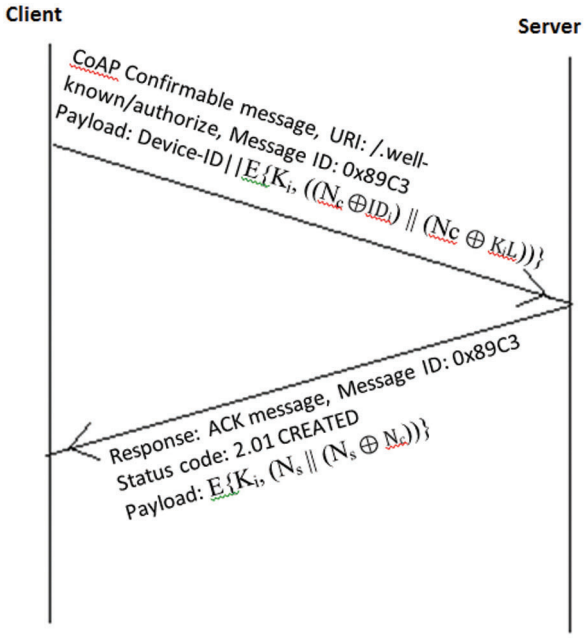

Figure 2: Two way handshake authentication between CoAP client and CoAP server

On reception of the message M1p from client i, the server will read the device identity IDi from the message and uses this 64 bit value to search the table kept in the server. The server will look for an entry in the table with received device ID. If the device ID is not found in the table, the client will be replied with the 4.01 unauthorized response. It will eliminate the burden of processing the message from a device which is not enrolled. Otherwise, if the device identity is found in the table, the associated shared key is fetched from the table.

$$P1 = \{DKi,\ C1\} \tag{2}$$

P1 is the 128 bit plaintext derived by decrypting C1 using the shared secret Ki

The most significant 64 bits of P1 is exclusive ORed with device identity of client IDi. It will return Nc if the message was received from genuine client.

$$Nc = IDi \oplus P1M \tag{3}$$

where P1M is the most significant 64 bits of P1. The Nc value is exclusive ORed with least significant 64 bits P1.

$$R1 = Nc \oplus P1L \tag{4}$$

where R is 64 bit value and P1L is the least significant 64 bits of plain text. Now the value R1 is compared with the least significant 64 bits of the secret key stored in server associated with the device identity IDi. If

this value is matched with the KiL value, then the client will be authenticated. A session will be created in the server to associate client with the server. This session resource will be created only after authenticating the client successfully in server side. If, value R1 does not match with the least significant 64 bits (KiL) of the shared secret key, the client will be replied with unauthorized message and no session will be created.

After the client is authenticated by server, the server will create a reply message for the confirmable message received from client. The status code of this reply message will be set to '2.01 created'. Server will create a nonce with the length of 64 bits long. The nonce created by server will be exclusive ORed with the nonce from client Nc. Server's nonce will be concatenated with the 64 bit value derived by concatenating client's nonce and server's nonce to generate 128 bit value. This 128 bit value is encrypted using the AES encryption key. This encrypted content is set as payload for the reply message to be sent to client i.

The payload of CoAP message is

$$M2p = E\ \{Ki,\ \ (Ns\ ||\ (Ns\ \oplus\ Nc))\} \tag{5}$$

where Ns is the 64 bit nonce generated by server. The message identity of response message is set to be same as the message identity of the confirmable message received from the client i.

On reception of CoAP message with payload M2p by client, it decrypts the message using the shared secret key Ki.

$$P2 = \{D\ Ki,\ M2p\} \tag{6}$$

P2 is the 128 bit plaintext derived by decrypting M2p.

Now the client will perform exclusive OR on the least significant 64 bits of P2 with the most significant 64 bits of P2.

$$R2 = P2L\ \oplus\ P2M \tag{7}$$

Result of the operation (R2) is compared with the nonce generated by the client (Nc). If R1 is matched with the value of Nc, then the server is authenticated by the client. Because only the intended server could have decrypted the message sent by client and sent back the nonce value in encrypted form.

After mutual authentication of client and server, the session key will be generated by using the nonce generated by the client and server. We have created the session key Ks as below.

$$Ks = Nc\ ||\ Ns \tag{8}$$

The server nonce and client nonce are concatenated to generate the 128 bit session key. This session key is valid for only the current session. At this stage of authentication the key Ks is known to both client and server. The data transmitted after authentication will be encrypted and decrypted using this session key. The AES key shared during manufacturing phase will be used for authentication purpose only which increases the security of the system.

4 Security Analysis of Proposed Scheme

The proposed technique for authentication is resistant against different known attacks. Here, various possibilities of attacks are assumed and showed that the proposed technique is reliable over these attacks.

Our proposed technique is resistant against identity theft. Assume that a malicious user has anyway got the device identity of a genuine client device, whose identity is already stored in server's table. The malicious user may send the device identity to server and the server will look for received device identity in stored table and will find a match. So it will perform decryption on the message payload using the associated secret key.

After decryption, the server will use this device id to extract a part of secret key from the decrypted payload by performing sequence of operations in the decrypted payload. If the payload was encrypted by the genuine user, server will be able to extract a part of shared key from the decrypted payload. Otherwise, in case of malicious user, server cannot extract a part of key from the decrypted message payload using device id sent by the malicious user.

Our proposed authentication technique provides confidentiality and is also resistant to man-in-the-middle attack where an unauthenticated malicious user enters into an online communication between two genuine users, remains escaped from the two parties. The attacker in the middle often monitors and changes information that was just realized by the two users. Assume that A_c and B_s are genuine client and genuine server. C_a is attacker in the middle of A_c and B_s. We assume that C_a is capable of capturing and modifying the messages between A_c and B_s. For authentication between client and server, the client A_c sends the session creation request to server. As C_a is capable of capturing messages, it can read the device identity of A_c. Even though, C_a has access to the message, it cannot read the nonce sent by the client A_c as the message is encrypted using AES. If the attacker tries brute force attack on the received encrypted payload, the attacker will be getting some predicted result T_k for every try k by using one key from set of 2^{128} keys. But the result T_k cannot be verified for the correctness of the nonce created by the client. To verify the correctness of predicted result T_k, the attacker will have to perform some additional steps for every T_k to extract nonce from the T_k. So for retrieving nonce, the attacker will not only have to break the proven AES algorithm but also have to undergo additional computational overhead. The attacker can succeed only if the AES key is successfully determined. Otherwise, even if the attacker has determined the nonce sent by the client, the attacker cannot get the nonce created by the server. Without knowing the nonce created by both client and server, it is not feasible to predict or determine the session key. So, the attacker C_a cannot modify or eavesdrop to the messages between client A_c and server B_s. Hence, our proposed authentication technique is secure against man-in-the-middle attack and provides confidentiality.

The proposed authentication technique also provides integrity to the messages between client A_c and server B_s.

Assume that $K = K_1K_2$

Where K is 128 bit AES key, K_1 is high order 64 bit of the K and K_2 is low order 64 bit of K.

M1 is message from A_c to B_s and M_2 is message from B_s to A_c. We assume that attacker C_a can capture messages M_1 and M_2. But the attacker cannot modify the payload of messages M_1 and M_2. If C_a modifies any of the payloads of the messages, the receiver can easily detect that someone has modified or the message is not from the intended sender. If the message M_1 was modified by C_a, B_s will be able to detect it. After reception of message M_1, B_s will decrypt the payload of the message M_1 using the secret key K. Then it will fetch the 64 bit device id from the payload of M_1. This Device id will be exclusive ORed with the high order 64 bits of the decrypted payload of M_1 after removing the device identity from the payload of M_1. The result of this exclusive OR operation will then be exclusive ORed with the low order 64 bits of the decrypted payload of message M_1 and this operation will result 64 bit value K_2. If the attacker had modified the message M_1 or M_1 was not created and encrypted by A_c, the above sequence of operations will not result K_2.

If the message M_2 from server B_c to client A_c was captured and modified by the attacker C_a, that can also be detected by the client A_c. Ac will decrypt the payload of message M_2 using the secret key K. Ac will then perform Exclusive OR operation between the high order 64 bit value and low order 64 bit value of the decrypted payload of message M_2. This operation will result the nonce that was created and sent to server in message M_1 by client. If the attacker had modified the message M_2 or M_2 was not created and encrypted by B_s, the above sequence of operations will not result the nonce created by client. This way, the senders of messages M1 and M2 can be authenticated and the integrity of these messages can be ascertained.

5 Results and Discussion

We have implemented the proposed technique for authentication using cooja simulator, a wireless sensor network simulator. We have used five client devices and a server device and these devices are emulated versions of Tmote Sky device which is a wireless sensor module from moteiv. Tmote sky is a MSP430 microcontroller based board with a cc2420 radio chip compatible to 802.15.4, 10 k RAM memory and 48k flash memory. UDP protocol is used as transport layer protocol and CoAP is used as application layer protocol. Fig. 3 shows the output window and Fig. 4 shows the network window of the cooja simulator. Output window shows the serial port output of all nodes in the network. Network window shows the network organization, all participating nodes, IPv6 addresses of each nodes, serial port output messages and lines to indicate the message transfer between clients and server. Node-1 is the server node and all other nodes are client nodes which will mutually authenticate with server node before data transmission.

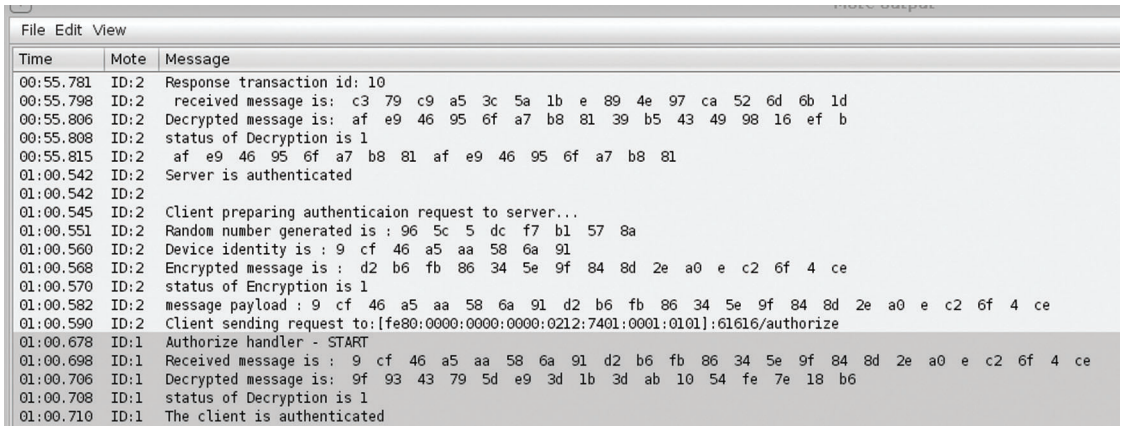

Figure 3: Output window

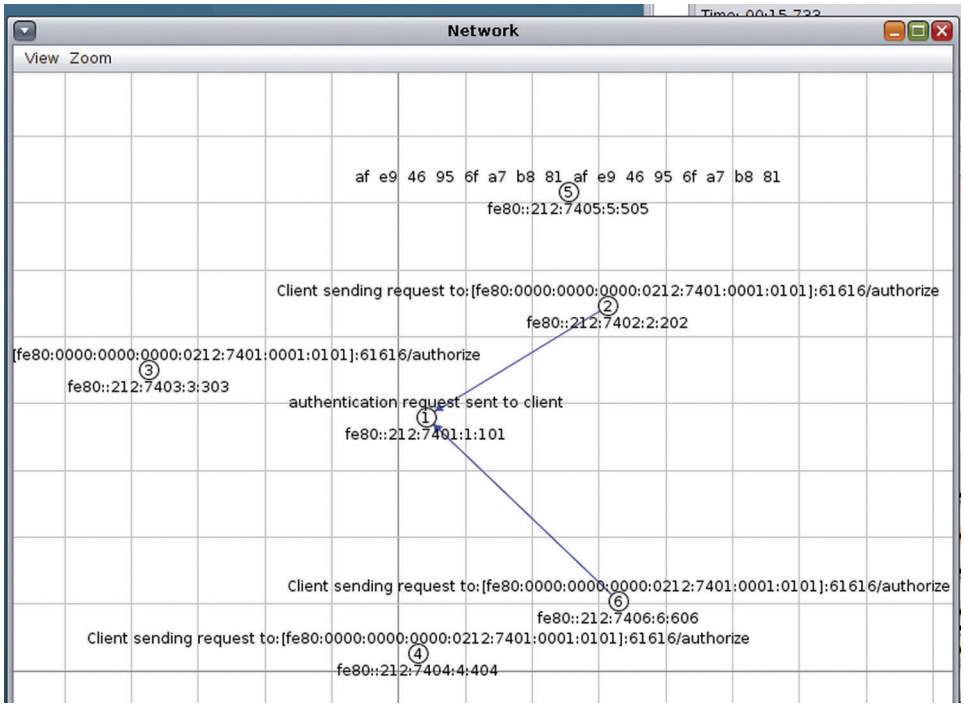

Figure 4: Network window

We have analyzed the performance of our proposed scheme against other authentication techniques proposed by Ukil et al. [10]. and Jan et al. [19,21] where authors have proposed authentication based on CoAP protocol. Various performance metrics like number of operations, client node processing time, transmit/receive hardware usage, packet size, number of hops and server response time were used to compare the performance of proposed technique with other techniques.

5.1 Number of Operations Performed

In our authentication scheme, very simple operations like exclusive-OR are used, which greatly reduces the processing overhead of the constrained devices. The operations and number of operations performed by client devices and server devices in various authentication schemes proposed by [15,19,21] were compared.

The Tab. 2 compares the number and type of operations executed by client and server devices of proposed scheme against three other authentication schemes which use CoAP payload authentication. Both client and server devices are performing only one AES encryption and only one decryption while other schemes perform three runs of AES algorithm. Executing the AES algorithm least number of times improves the performance of the client and server and it also improves the power consumption of client and server devices.

Table 2: Comparison of number and type of operations performed in various authentication schemes

Operations	Our proposed scheme		Authentication proposed by Jan et al. in [21]		Authentication proposed by Ukil et al. in [15]		Authentication proposed by Jan et al. in [19]	
	Client	Server	Client	Server	Client	Server	Client	Server
AES 128 bit encryption	1	1	1	2	1	2	1	2
AES 128 bit decryption	1	1	2	1	2	1	2	1
128 bit Ex-OR	2	2	2	2	2	2	2	2
No. of CoAP messages handled (sent/ Received)	2	2	4	4	4	4	4	4
Session creation	-	1	-	1	-	1	-	1

5.2 Node Processing Time

The time taken by client node to complete the mutual authentication is considered for comparing processing time. This time includes the time taken to prepare CoAP message, AES encryption, AES decryption and other operations except the message transmission time. The processing times of each of the five client nodes were determined using the timeline window of cooja simulator. Average of these values is used as the processing time of client node. Fig. 5 shows the comparison chart for client node processing time of proposed technique against other similar authentication schemes. The proposed technique roughly halves the client node processing time taken by other mutual authentication schemes.

5.3 Transmit and Receive Hardware Utilization

The client node, while sending authentication request message to server and receiving response from server, will turn on the hardware for transmission and reception. This hardware ON duration will have direct impact on the power consumption of battery operated devices and low power constrained network

devices. Cooja timeline window can be used to determine the hardware ON time in client nodes and in server node. Fig. 6 compares the duration for which the transmitting and receiving hardware of client node is kept ON during authentication in proposed technique and in similar other authentication techniques. As shown in the graph, the hardware utilization in proposed scheme halves the hardware ON time in compared authentication schemes.

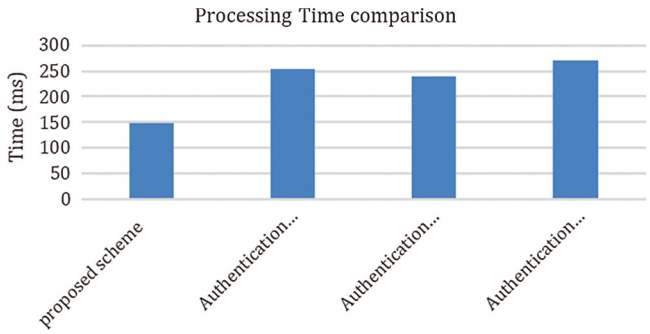

Figure 5: Client node processing time comparison

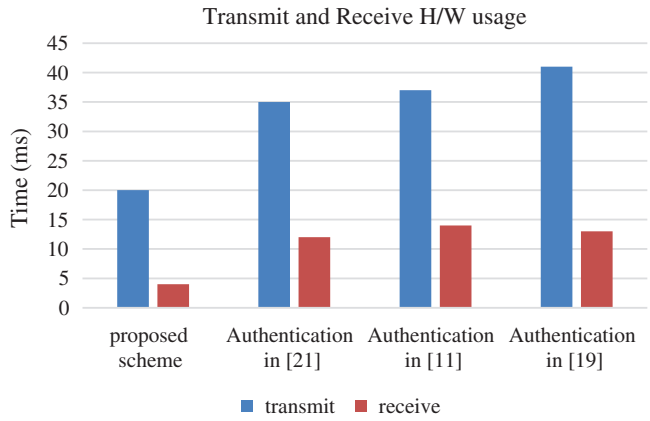

Figure 6: Hardware ON time comparison

5.4 Server Node Response Time

We tested the time taken by server node to process the client request for different number of client nodes. This server response time includes the time taken to process the authentication request message from client, verifying the genuineness of client node and preparing authentication request message to client. The server response time is calculated from immediately after the reception of authentication request message from client to the start of sending authentication message to client. We assessed the response time of server in different test environment with 1, 5, 10 and 15 client nodes. In each of the test environment, we allowed client nodes to simultaneously generate authentication request to server and assessed the response time for every authentication request. Then, the average response time is calculated by taking average of all these readings.

If N client nodes are in the network, the average response time (R_a) of server node is determined by

$R_a = \text{sum}(t_1, t_2, \ldots t_N)/N$ where t_i is the time taken by server node to respond node i.

Tab. 3 shows the average response time assessed for N = 1, 5, 10 and 15.

Table 3: Average server response time

Number of client nodes	1	5	10	15
Average response time (ms)	78	165	211	284

5.5 Packet Size and Number of Hops

The size of the message and number of messages transmitted between authenticating devices play important role in power consumption and network traffic especially in constrained devices network. Our proposed scheme need to manipulate only two CoAP messages in client device and two CoAP messages in server device, whereas other schemes manipulate four CoAP messages in client device and four CoAP messages in server device. The comparison chart in Fig. 7 depicts the packet size and number of authentication messages transmitted between client and server in proposed scheme and in other schemes considered for comparison. The payload of the authentication request CoAP message occupies 24 bytes and payload of response CoAP message from server occupies 16 bytes in proposed scheme, whereas other schemes occupy 32 bytes of payload in each of the CoAP messages.

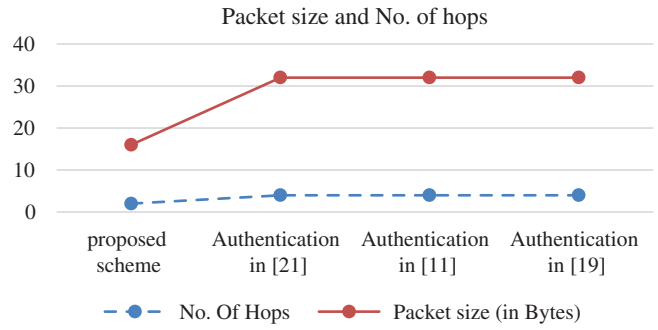

Figure 7: Packet size and number of hops comparison

In our proposed scheme, number of messages exchanged between IoT devices for the accomplishment of authentication is two only. In other authentication techniques, the number of messages exchanged to complete handshaking between client and server is four. Hence, the obvious advantage in performance of proposed scheme is reduction in round trip time as it takes only one round trip time. It reduces 50% of the total time required to exchange messages between client and server by the authentication techniques compared with our proposed technique. This message transmission time does not include the time taken by client and server for processing messages.

6 Conclusion

In this paper, we have proposed a technique for authentication which is lightweight in terms of processing and network overhead as we have used lightweight protocols, lightweight algorithms and lightweight operations. In this technique, client and server share a common key in manufacturing and deployment phase. The server and client authenticates mutually if the client needs to observe for any resources in the server. For authentication, client and server use advanced encryption standard algorithm with the pre-shared secret key for securing messages between server and client. Once the authentication is completed, a unique identity will be created in the server for the client. Further communication will be encrypted using the session key exchanged between client and server during authentication.

Our scheme is good for the devices with very limited memory and processing resources. We have reduced the number of messages exchanged between client and server during mutual authentication between them. Reduction of number of messages improves the efficiency of network utilization, time taken for authentication and power consumption by communicating devices. We have also shown that our scheme has greater advantage of reducing the processing overhead in the resource constrained client device and server device. The length of the random nonce used for authentication can be increased to improve the security of the scheme based on the devices used for communication and the requirement of the application. In future new authentication protocol can be used for better security.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] P. Datta and B. Sharma, "A survey on IoT architectures, protocols, security and smart city based applications," in *Proc. ICCCNT*, IEEE, Delhi, India, pp. 1–5, 2017.

[2] K. L. Lueth, "Iot analytics: Why the internet of things is called internet of things: Definition, history, disambiguation," 2014. Online Available: https://iot-analytics.com/internet-of-things-definition/.

[3] D. Minoli, K. Sohraby and J. Kouns, "Iot security (IoTSec) considerations, requirements, and architectures," in *Proc. CCNC*, IEEE, Las Vegas, NV, USA, pp. 1006–1007, 2017.

[4] R. Fielding and J. Reschke, "Hypertext transfer protocol (http/1.1): Semantics and content," ed: rfc 7231, June, 2014.

[5] Z. Shelby, "Embedded web services," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 52–57, 2010.

[6] T. Levä, O. Mazhelis and H. Suomi, "Comparing the cost-efficiency of CoAP and HTTP in web of things applications," *Decision Support Systems*, vol. 63, pp. 23–38, 2014.

[7] W. Colitti, K. Steenhaut and N. De Caro, "Integrating wireless sensor networks with the web," In *Workshop Proceedings of Extending the Internet to Low power and Lossy Networks IP + SN*, Chicago IL USA, 2011.

[8] Z. Shelby, K. Hartke and C. Bormann, "The constrained application protocol (CoAP)," From draft-ietf-core-coap-18, In RFC-7252, June 2014, Available online at: https://datatracker.ietf.org/doc/html/rfc7252.

[9] D. Trabalza, S. Raza and T. Voigt, "Indigo: Secure coap for smartphones," *in Wireless Sensor Networks for Developing Countries*, 1st edition. , Springer, Cham, vol. 366, pp. 108–119, 2013.

[10] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, A. Pal and T. Bose, "Lightweight security scheme for IoT applications using CoAP," *International Journal of Pervasive Computing and Communications*, vol. 10, no. 4, pp. 372–392, 2014.

[11] K. Hartke and O. Bergmann, "Datagram transport layer security in constrained environments," in CoRE Working Group, Internet-draft, draft-hartke-core-codtls-02, IETF, July 16, 2012, Available online at: https://tools.ietf.org/pdf/draft-hartke-core-codtls-02.pdf.

[12] S. Raza, H. Shafagh and O. Dupont, "Compression of record and handshake headers for constrained environments," DICE Working Group, Internet-draft, draft-raza-dice-compressed-dtls-00, March 2014. Available online at: https://www.ietf.org/archive/id/draft-raza-dice-compressed-dtls-00.pdf.

[13] T. Freeman, R. Housley, A. Malpani, D. Cooper and W. Polk, "Server-based certificate validation protocol (SCVP)," in Network Working Group, RFC 5055, December 2007. Available online at: https://www.rfc-editor.org/rfc/pdfrfc/rfc5055.txt.pdf.

[14] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig and G. Carle, "DTLS based security and two-way authentication for the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2710–2723, 2013.

[15] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, A. Pal and T. Bose, "Auth-lite: lightweight m2mauthentication reinforcing DTLS for CoAp," in *Proc. PERCOM*, IEEE, Budapest, Hungary, pp. 215–219, 2014.

[16] J. Park and N. Kang, "Lightweight secure communication for CoAP-enabled internet of things using delegated DTLS handshake," in *Proc. ICTC*, IEEE, Busan, Korea (South), pp. 28–33, 2014.

[17] J. Granjal, E. Monteiro and J. S. Silva, "On the effectiveness of end-to-end security for internet-integrated sensing applications," in *Proc. ICGCC*, IEEE, Besancon, France, pp. 87–93, 2012.

[18] T. Shah and S. Venkatesan, "Authentication of IoT device and IoT server using secure vaults," in *Proc. TrustCom/BigDataSE*, IEEE, New York, NY, USA, pp. 819–824, 2018.

[19] M. A. Jan, F. Khan, M. Alam and M. Usman, "A payload-based mutual authentication scheme for internet of things," *Future Generation Computer Systems*, vol. 92, pp. 1028–1039, 2019.

[20] S. Arvind and V. A. Narayanan, "An overview of security in coap: Attack and analysis," in *Proc. ICACCS*, IEEE, Coimbatore, India, pp. 655–660, 2019.

[21] M. A. Jan, P. Nanda, X. He, Z. Tan and R. P. Liu, "A robust authentication scheme for observing resources in the internet of things environment," in *Proc. TrustCom*, IEEE, Beijing, China, pp. 205–211, 2014.

[22] M. Shanmugam and R. Asokan, "A machine-vision-based real-time sensor system to control weeds in agricultural fields," *Sensor Letters*, vol. 13, no. 6, pp. 489–495, 2015.

[23] M. Shanmugam and A. Ramasamy, "Sensor-based turmeric finger growth characteristics monitoring using embedded system under soil," *International Journal of Distributed Sensor Networks*, vol. 10, no. 6, pp. 476176, 2014.

[24] M. A. Bahnasawi, K. Ibrahim, A. Mohamed, M. Khalifa Mohamed, A. Moustafa *et al.*, "ASIC-Oriented comparative review of hardware security algorithms for internet of things applications," in *Proc. ICM*, IEEE, Giza, Egypt, pp. 285–288, 2016.

[25] E. Rescorla and B. Korver, "Guidelines for writing RFC text on security considerations BCP 72," RFC 3552, 2003. [Online] Available: https://www.hjp.at/doc/rfc/rfc3552.html.