# An Enhanced Scheme for Authentication Using OTP and QR code for MQTT Protocol

**Arvind P. Jayan, Ashwin Balasubramani, Akshay Kaikottil and N. Harini***

***Abstract--- The pervasive nature of computing makes classical solutions inapplicable for handling communication between peers in the IOT platform. With enormous number of users and devices becoming a part of this network, Authentication of communication devices has become more challenging. The work presented in this paper discusses an authentication scheme with reduced overhead based on multi factors, barcodes and One Time Password. The outcome of the experimentation revealed the efficiency of the scheme in terms of guaranteeing enhanced security without compromise in versatility.***

***Keywords--- MQTT (Message Queuing Telemetry Protocol), IoT (Internet of Things), AES (Advanced Encryption Standard), sniffing, authentication, QR code (Quick Response Code).***

## I. INTRODUCTION

Computing has now become pervasive. The existence of many devices to sense, collect and process data from around the world has necessitated the need to look into how these devices could securely communicate with each other. A bar code is an optical exemplification of data in a machine-readable form generally representing data by varying the width and the space information between horizontal and vertical lines. A special type of bar code called Quick Response Code facilitates 360 degree reading. This work examines the suitability of the QR code in terms of authenticating the peers connected in an IOT network using MQTT protocol. The results of experimentation clearly revealed the suitability of the bar codes in enhancing the security between communicating peers over other methods like notifying and receiving an SMS-OTP etc. The method also was found to be cheaper as it does not require GSM. To ensure uninterrupted and authorized services many password/smart cards based and/or hybrid schemes have been proposed in literature for multi-server architecture. Although these schemes work well, the overhead involved in the process is high. The paper has tried to integrate symmetric key encryption, hash functions and bar codes with the objective of decreasing the overhead associated with the authentication. [6]-[7]

## II. LITERATURE REVIEW

Rapid evolution in information communication technology force digital communication between peers in IOT. This device to device communication is generally based on either pushing or polling protocol. The light weightiness and high productivity factors make push protocols more suitable for IOT devices. Widely used push protocols include XMP, MQTT etc. [4]

*MQTT Protocol*

Amongst all the light weight messaging protocols MQTT (Message Queuing Telemetry Transport) protocol is known for its simple methodology. It is designed to ease the communication between network clients for distributing telemetry information. This machine-to-machine (M2M) communication is achieved via a publisher/subscriber pattern. The resource constrained IOT devices send/publish information about any particular topic with the help of a 'Broker'. The Broker in turn is responsible for sending the right information to the right subscriber that has subscribed that particular topic. MQTT protocol is a dependable choice for wireless networks that experience varying latency levels due to bandwidth changes and unreliable connections since the protocol doesn't require additional routing or networking features. It also provides a light weight, open and simple data transportation service. A security analysis on the MQTT protocol revealed the fact that packet information is communicated in plain text. This would undoubtedly affect the security guarantees of the protocol. As the published messages are in plaintext, attackers could interfere with the communications between subscriber and publisher devices and the also the Brokers. Another drawback with this protocol is its incapability to prioritize requests. [9]-[12]

*MQTTLens*

MQTT Lens is a chrome extension available with the browser for the purpose of connecting to an MQTT Broker and test with publish/subscribe operations. This tool could be used as a client to communicate with the

Manuscript received February 01, 2019

**Arvind P. Jayan,** Dept of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Tamil Nadu, Amrita Vishwa Vidyapeetham, India. (e-mail: cb.en.u4cse15007@cb.students.amrita.edu)

**Ashwin Balasubramani,** Dept of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Tamil Nadu, Amrita Vishwa Vidyapeetham, India. (e-mail: cb.en.u4cse15008@cb.students.amrita.edu)

**Akshay Kaikottil,** Dept of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Tamil Nadu, Amrita Vishwa Vidyapeetham, India. (e-mail: cb.en.u4cse15002@cb.students.amrita.edu)

**N. Harini*,** Dept of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Tamil Nadu, Amrita Vishwa Vidyapeetham, India. (e-mail: n_harini@cb.amrita.edu)

broker.

*Sniffing Tools*

A device used for monitoring the inbound and outbound packets in a network. This tool will help in research and security by capturing and visualizing the packets in the network traffic.

*Wireshark*

A tool that analyses the packet for troubleshooting network related issues. The tool works on multiple platforms (UNIX, Solaris, Microsoft Windows, etc) and is generic. The tool uses ECAP to capture packet information and enables users to put their network in promiscuous mode to monitor the broadcast or multicast traffic. The tool also has the capability of understanding the structure of different networking protocol. [14]

*Symmetric Key Cryptography*

The crypto systems are classified into Block Ciphers and Stream Ciphers in order to process the data in digital form, which are being represented as continuous strings of binary digits. Block ciphers unlike Stream ciphers have a fixed number of blocks that are used for processing the plain binary text; i.e. a set of operations specific to blocks, are performed on plain text blocks for generating a block of ciphertext bits. Symmetric key encryption which is also known as secret key encryption, is the process where both encryption and decryption process is done using one key. Reports state that Symmetric encryption techniques generally require less processing power for computation. Hence, they are faster than Asymmetric encryption (a method that uses two keys). Based on symmetric key cryptography the best examples of Block Ciphers are Data Encryption Standard (DES) and Advanced Encryption Standard (AES).

AES was initially recommended by NIST to replace DES as it was very low in terms of key size.[5],[10]

*Advanced Encryption Standard (AES)*

AES methodology is purely based on the 'Substitution-Permutation Network'. It implements all of its computational work in bytes.

The 128 bit block of plaintext is considered as a group of 16 bytes.

The bytes of data are further classified and ordered into rows and columns, four in number and represented as a matrix.

This matrix is then processed using a series of operations such as substitution, shifting, mixing etc, to produce the ciphertext.

The decryption operations are a reversing of the encryption process. The working of AES scheme is illustrated in the figure 1. [2]

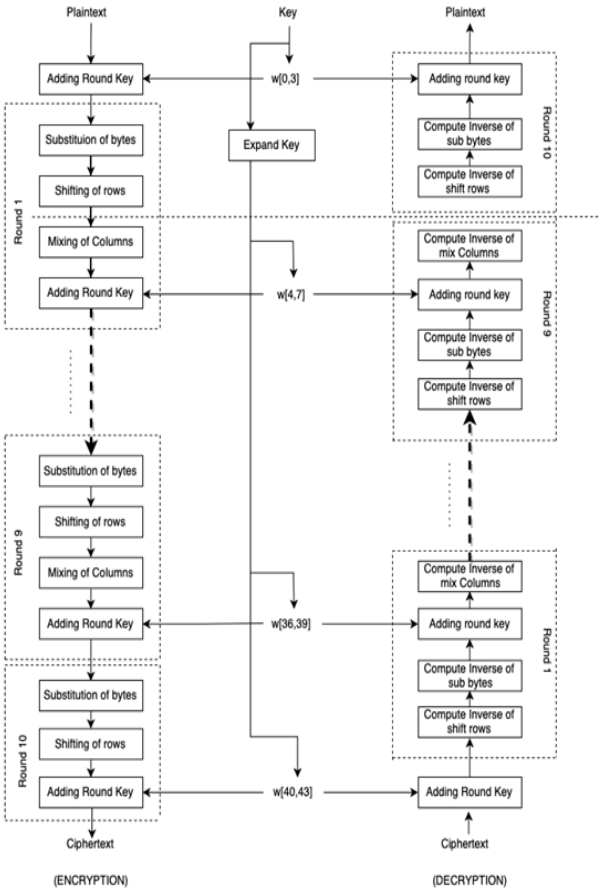


Fig 1: Architecture of Advanced Encryption Standard

*Hashing*

The term 'Hash' in the context of cryptography refers to a function that produces a fixed length random string for any variable length input text. The function satisfies properties like one-wayness, weak collision resistance and strong collision resistance. Hash functions find their applicability in digital signatures, digital time stamping, digital steganography, etc. Attacking a hash functions generally means breaking one of these properties. A brute force attack works on all hash functions. The complexity in crypt analysis of hash functions is generally introduced by the structure of the hash function and/or on the algorithm of compression function. Family of hash functions are being used by real time applications for verifying data integrity. SHA 256 is considered to be one of the strongest hash functions available under SHA family [8]

*SHA 256*

Secure Hash Algorithm (SHA-256) is a keyless hash function with the digest length of 256 bits. It is an MDC (Manipulation Detection Code), that is, a keyless hash function [11]. The computation is depicted in Figure 2.

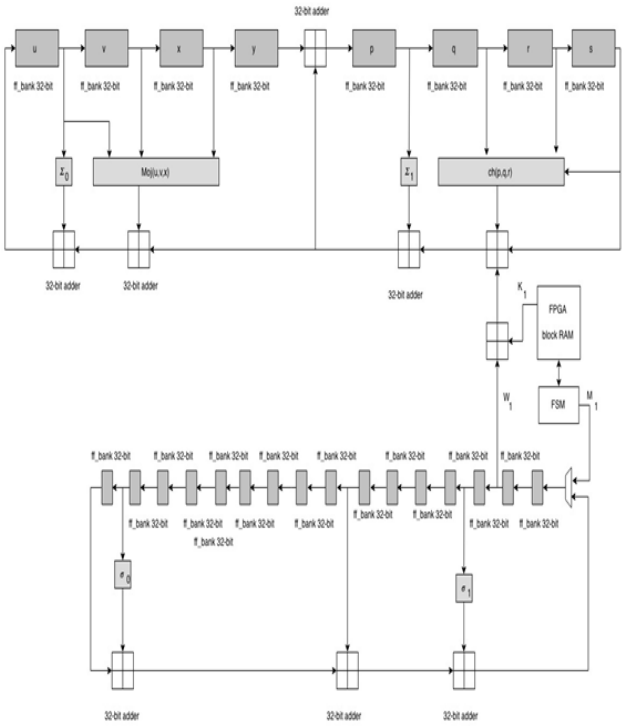


**Fig 2: Architecture of Secure Hash Algorithm-256**

*QR Code*

A QR code [Fig 3], which is also called as a Matrix code, is a machine-readable encoding of information where the data is said to be two-dimensional. It consists of 360 degree readable black and white squares. The Quick Response Code can be stored in many forms such as contact information, photo and video links, plain text and many more. It is also known to be stored as a URL. The storage level of QR codes can range up to 7,089 characters of information, which is very large when it is put in comparison with 1-D barcode. The encodable character set of QR code comprises of Numbers (0-9), Alphabets (uppercase A-Z), Nine Special characters (% * + - / _ $) and Kanji characters. [1]

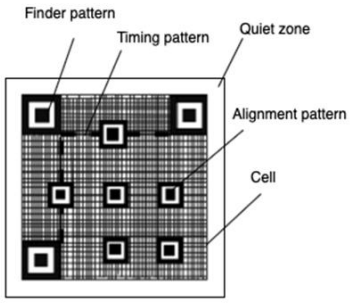


**Fig 3: Quick Response Code**

*Authentication*

Authentication is done in cryptography in order to verify the originator of the information. It is also an important aspect of gaining assurance with respect to parameters of data such as time and date. Authentication and Key agreement protocols are fundamental building blocks for securing communications over a network.

Among the available authentication schemes Password based authentication schemes is particularly attractive due to its unique features. To improve the security levels, variations like password chaining, one-time passwords are used in real-time systems. With IOT becoming popular Bar Code based authentication is also gaining popularity. The fact that 3D bar codes are capable of storing huge volumes of data enables multiple factors could be combined and stored in bar codes to enhance security. [13]

*Summary of Findings*

Rapid development in information and network technologies has led to the emergence of new computing paradigms. With enormous number of devices being added to the network it is important for one to carry out a thorough security analysis. Although literature discusses many authentication schemes based on single as well as multiple factors, a thorough analysis revealed that the schemes involve high overhead in computations which make it less desirable for practical applications. This brings out a clear need for a new usable authentication scheme capable of minimizing the computational overhead.

*Problem Statement*

To enhance the security of peers in terms of minimizing computational overhead for devices in IOT network communicating using MQTT protocol.

## III. PROPOSED SYSTEM

The scheme works in three phases.

1. *Registration phase:*

Initially, the Clients in the network must register details regarding their IP Address, MAC Address in the Registration Centre [Fig 4.1]. The Registration Centre is the central authority for assigning symmetric keys for the publisher and subscriber to encrypt and decrypt the details respectively. The symmetric key is used by the sender to encrypt the data and by the receiver to decrypt it. It also assigns the Broker with a digital signature key in order to verify the originality of the publisher and subscriber. [3]

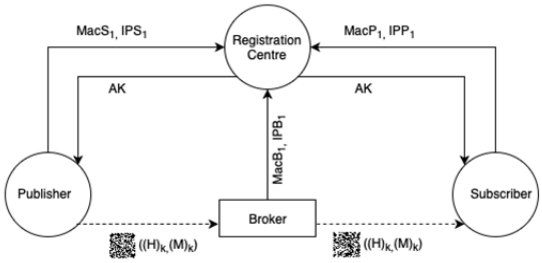


**Fig 4.1: Registration Phase**


Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

*2. Login Phase:*

In the Login phase [Fig 4.2] the publisher encrypts the IP Address of both the subscriber and publisher, MAC Address of the same, Time Stamp and Random number using symmetric key algorithm. Time stamp represents the time at which the packet is being sent. A single random number is also generated for both the clients. These details are also hashed using a hash function. The encrypted information and the hashed information are concatenated and encoded in a QR Code. QR Code is communicated to the subscriber as an OTP message. Meanwhile, the Broker verifies the subscriber's identity with a digital signature algorithm initially assigned by the Registration Centre.

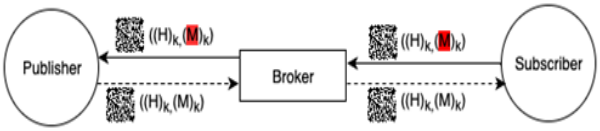


**Fig 4.2: Login Phase**

*3. Verification Phase:*

The QR Code that has been sent by the publisher is decoded by the subscriber and then the data thus obtained is decrypted. The decrypted OTP message is then hashed by a Hash function. The result is then compared with the received hash value. A match of the hash values indicate successful [Fig. 4.3.1] authentication. If the Hash values are not matching, then the authentication is a failure [Fig. 4.3.2]. The subscriber then sends the decrypted message back to the publisher in order to verify its identity.

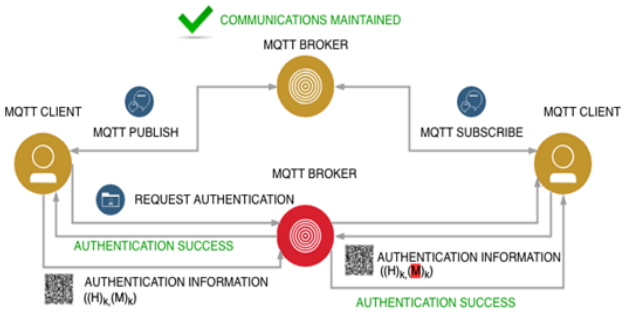


**Fig 4.3.1: Verification phase showing successful authentication**

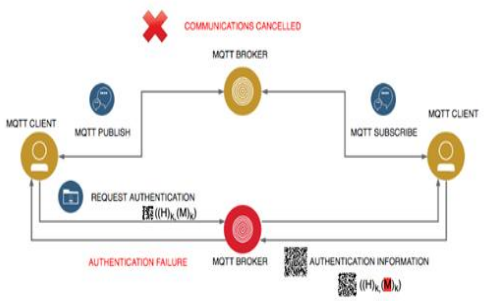


**Fig 4.3.2: Verification phase showing authentication failure**

**Table 1: Notations**

| Symbol | Meaning |
|---|---|
| $IPP_i$ | IP Address of the $i^{th}$ Publisher |
| $IPS_i$ | IP Address of the $i^{th}$ Subscriber |
| $IPB_i$ | IP Address of the $i^{th}$ Broker |
| $MacP_i$ | Mac Address of the $i^{th}$ Publisher |
| $MacS_i$ | Mac Address of the $i^{th}$ Subscriber |
| $MacB_i$ | Mac Address of the $i^{th}$ Broker |
| AK | AES Encryption Key |
| | QR Code |
| $(H)_k$ | Hashing using SHA-256 |
| $(M)_k$ | Message encrypted by AES |
| $(M)_k$ | Message decrypted by AES |

## IV. RESULTS AND DISCUSSIONS

A closed network was setup involving subscribers and publishers whose communication was facilitated through 'Hivemq' Broker. On monitoring the network with a sniffing tool Wireshark, it was observed that the contents of the packets were transmitted in plain text. As a first step to improving security, individual packets were encrypted using public key algorithms RSA, ECC. This method was found to involve more overhead and it also had a shortcoming that if the login credential was compromised by an adversary then he/she could tamper the system. With the aim of minimizing the overhead involved in the authentication process, the proposed scheme was modeled to carry out regular or frequent handshakes between the publisher and the subscriber using a bar code (QR Code) technique. This was fruitful because of the short nature of the key and the encoding process which is light weighted in nature than an encryption process with a larger sized key. The snippets of the code and screenshots of the results are depicted in figure 5.1 - 5.5

Fig 5.1 provides the decrypted information of the QR code that is received. The decrypted text is hashed and compared with the hash that has been received. Fig 5.2 depicts the QR code that is received for authentication. Fig 5.3 and Fig 5.4 illustrates the code snippets for both publisher and subscriber. Fig 5.5 and 5.6 represent the communication of messages through MQTT lens respectively.

Fig 5.7 illustrates the code snippets for QR code generation.

```
values obtained from scanning qrcode:  b'gAAAAABcDgQkXbCEJ0c16yMiOtaRcKk3YbsbIid
bl6ekcrvHc8eMwgNUMU8VUYTlZUBdZUHkQe8oGj2sm-9ENYbopMgHt82nfW2M8sdZ6LKHjlx_U1MskfB
zbPG7RTAVGFLwT6k9gpXEInHQOsA12SQHhL0CK_FDyvw--wJ8je-BcRHffZiqCU9FRfqZlFuC4ba-Oli
hXoXl\x1a\xa6\xc9\x86p \x1bCq6k}\xb8\x90@\x0e\x9f=:\xcb\xdf\xfa\x91>=\xb1\xf3%\x
ba4\x0bi'

Decrypting encrypted text:  b'127.0.1.1,8796758373432,127.12.3.3,12:22:33:33,201
8-12-10 11:43:56.882258,933491'

Hashing the values for verification

Authentication Successful
```

**Fig. 5.1: Decryption of the received data**

**Fig 5.2: QR Code output**

```python
import paho.mqtt.client as mqttClient
import time

def on_connect(client, userdata, flags, rc):

    if rc == 0:

        print("Connected to broker")

        global Connected                #Use global variable
        Connected = True                #Signal connection

    else:

        print("Connection failed")

def on_message(client, userdata, message):
    print ("Message received: "  + str(message.payload))

Connected = False   #global variable for the state of the connection

broker_address= "broker.hivemq.com"  #Broker address
port = 1883                          #Broker port
user = "APJ"                         #Connection username
password = "password"                #Connection password

client = mqttClient.Client("Python")               #create new instance
client.username_pw_set(user, password=password)    #set username and password
client.on_connect= on_connect                      #attach function to callback
client.on_message= on_message                      #attach function to callback

client.connect(broker_address, port=port)          #connect to broker

client.loop_start()        #start the loop

while Connected != True:    #Wait for connection
    time.sleep(0.1)

client.subscribe("python/test")

try:
    while True:
        time.sleep(1)

except KeyboardInterrupt:
    print ("exiting")
    client.disconnect()
    client.loop_stop()
```

**Fig 5.3 Snapshot of Subscriber Code Snippet**

```python
import paho.mqtt.client as mqttClient
import time

def on_connect(client, userdata, flags, rc):

    if rc == 0:

        print("Connected to broker")

        global Connected                #Use global variable
        Connected = True                #Signal connection

    else:

        print("Connection failed")

Connected = False   #global variable for the state of the connection

broker_address= "broker.hivemq.com"
port = 1883
user = "APJ"
password = "password"

client = mqttClient.Client("Python")               #create new instance
client.username_pw_set(user, password=password)    #set username and password
client.on_connect= on_connect                      #attach function to callback
client.connect(broker_address, port=port)          #connect to broker

client.loop_start()        #start the loop

while Connected != True:    #Wait for connection
    time.sleep(0.1)

try:
    while True:

        value= raw_input("Enter message: ")
        client.publish("python/test",value)

except KeyboardInterrupt:

    print("exiting")
    client.disconnect()
    client.loop_stop()
```

**Fig 5.4 Snapshot of Publisher Code Snippet**

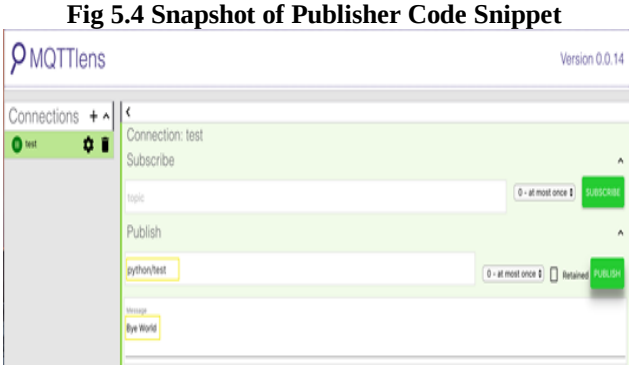


**Fig 5.5 subscribing a message via MQTT Lens**

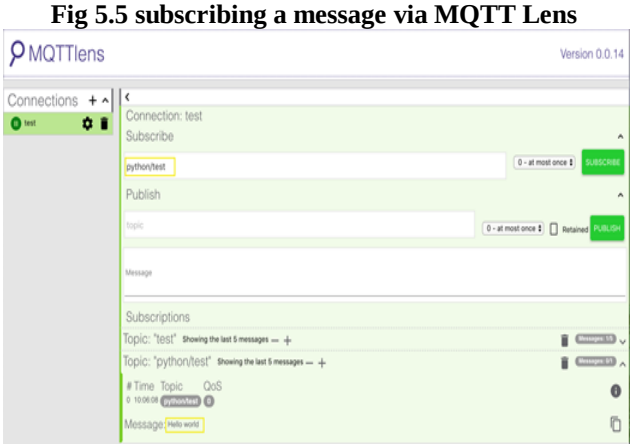


**Fig 5.6 publishing a message via MQTT Lens**


*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication*

```
qrgen.py
from __future__ import print_function
import qrcode
from datetime import datetime
from uuid import getnode as get_mac
import socket
import pyotp
import hashlib
from base64 import urlsafe_b64encode, urlsafe_b64decode
from cryptography.fernet import Fernet

hostname=socket.gethostname()
IPAddr=socket.gethostbyname(hostname)
mac = get_mac()
mac=str(mac)
recvip,recvmac=input('Enter the reciever\'s IP address,MAC address:').split()
#recvhostname=str(recvhostname)

totp=pyotp.TOTP("JBSWY3DPEHPK3PXP")
otp=str(totp.now())

recvip=str(recvip)
recvmac=str(recvmac)
finalstring=IPAddr+','+mac+','+recvip+','+recvmac+','+str(datetime.now())+','+ otp
hashfs=hashlib.sha256(finalstring.encode())
print('\n\nThe concatenated string is:',finalstring)
print('\nThe hashed string is: ',hashfs.digest())

message=finalstring

key = Fernet.generate_key()
cipher_suite = Fernet(key)
cipher_text = cipher_suite.encrypt(message.encode())
print("\ncipher text, encrypted using symmetric key:",cipher_text)

sig=cipher_text+hashfs.digest()
img=qrcode.make(sig)
img.save('image.png')
print("\nqrcode generated successfully")

plain_text = cipher_suite.decrypt(cipher_text)
print("\n\n\nvalues obtained from scanning qrcode: ",sig)
print("\nDecrypting encrypted text: ", plain_text)
print("\nHashing the values for verification")
hashrs=hashlib.sha256(plain_text)
if hashrs.digest()==hashfs.digest():
    print("\n Authentication Successful")
else:
    print("\n Authentication Unsuccessful")
```

**Fig 5.7 Snapshot of QR code generation**

## V. CONCLUSION AND SCOPE FOR FUTURE

In this paper, an authentication scheme that is capable of operating with reduced overhead, applicable for IOT devices communicating using MQTT protocol has been proposed. As a future extension to it, it is planned to test the scheme to evaluate its capability in terms of mitigating cryptographic attacks like man in the middle, replay, and phishing.

## REFERENCES


1. Kinjal H. Pandya and Hiren J. Galiyawala.(2014). A Survey on QR Codes: in context of Research and Application. International Journal of Emerging Technology and Advanced Engineering,. 4(3): 258-262.
2. Shyamala C.K., Harini N. and Padmanabhan T.R. (2011) Cryptography and security, First Edition, Wiley India, 560p.
3. Kamakshi Devisetty R N, Aruna D. and Harini.N. (2018) Secure Proxy Blind ECDS Algorithm for IoT‖, International Journal of Pure and Applied Mathematics. 118 (7): 437-445.
4. Sklavos, Nicolas and Zaharakis, I. (2016). Cryptography and Security in Internet of Things (IoTs): Models, Schemes, and Implementations. *8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Larnaca, pp. 1-2. DOI: 10.1109/NTMS.2016.7792443
5. Singh, G. (2013). A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. International Journal of Computer Applications, 67(19): 33-38.
6. Lee, Young-Sil, Hyun Kim, Nack, Lim, Hyotaek, HeungKuk, and Lee Hoonjae. (2010) Online banking authentication system using mobile-OTP with QR-code. *5th International Conference on Computer Sciences and Convergence Information Technology*, Seoul, pp. 644-648. DOI: 10.1109/ICCIT.2010.5711134.
7. Oh DS., Kim BH., Lee JK. (2011) A Study on Authentication System Using QR Code for Mobile Cloud Computing Environment. In: Park J.J., Yang L.T., Lee C. (eds) Future Information Technology. Communications in Computer and Information Science, Vol 184. Springer, Berlin, Heidelberg. Pp.500-507.
8. Preneel B. (2010) Cryptographic Hash Functions: Theory and Practice. In: Soriano M., Qing S., López J. (eds) Information and Communications Security. ICICS 2010. Lecture Notes in Computer Science, vol 6476. Springer, Berlin, Heidelberg. Pp. 1-3.
9. Yassein, M. B., Shatnawi, M. Q. Aljwarneh S. and R. Al-Hatmi. (2017) Internet of Things: Survey and open issues of MQTT protocol," *International Conference on Engineering & MIS (ICEMIS)*, Monastir, pp. 1-6. DOI: 10.1109/ICEMIS.2017.8273112
10. Stallings, W. (2006). Cryptography and Network Security: Principles and practices. Fifth Edition. Pearson Education, India. 900p.
11. Gowthaman, A and Manickam, Sumathi. (2015), Performance study of enhanced SHA-256 algorithm. International Journal of Applied Engineering Research 10 (4):10921-10932.
12. Soni, Dipa and Makwana, Ashwin. (2017). A survey on MQTT: a protocol of Internet of Things (IOT). International conference on telecommunication, power analysis and computing techniques (ICTPACT - 2017), Chennai, India.
13. Syed Zulkarnain Syed Idrus, Estelle Cherrier, Christophe Rosenberger and Jean-Jacques Schwartzmann. (2013). A Review on Authentication Methods. Australian Journal of Basic and Applied Sciences,7(5):95-107.
14. Qadeer, M.A., A. Iqbal, M. Zahid and M. R. Siddiqui (2010) Network Traffic Analysis and Intrusion Detection Using Packet Sniffer. *Second International Conference on Communication Software and Networks*, Singapore, pp. 313-317.




*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication*