Journal of Theoretical and Applied Information Technology
15th January 2019. Vol.97. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

LIGHTWEIGHT SECURE SCHEME FOR IOT-CLOUD CONVERGENCE BASED ON ELLIPTIC CURVE

[1]AMRANI AYOUB, [2]RAFALIA NAJAT, [3]ABOUCHABAKA JAAFAR
[1,2,3]Computer and Telecommunications Research Laboratory
E-mail: [1]amrani.ayoub@uit.ac.ma, [2]najat.rafalia@uit.ac.ma, [3]abouchabaka@uit.ac.ma

ABSTRACT

The internet of things appears as a solution to connect people around the world. Its utility lies in the ability to connect objects and exchange information anywhere and everywhere. Many objects and services in different fields will be created, such as smart homes, e health, transport and logistics. The evolution of IoT, increases the number of connected object that generates a huge number of data. However, with the low capacity of storage and processing of these objects, there is a requirement to connect these objects to a large pool of resource like Cloud computing. The convergence between IoT and Cloud, will bring many services that will be of great benefit to humanity. However, this convergence will not see the day unless the communication between devices and the Cloud is secure. Most of the secure scheme proposed, that we will quote in the following sections, either have a weakness on their scheme, or are based on Hypertext Transfer Protocol (HTTP) which consumes bandwidth and which will exhaust the resources of the devices. Publish / Subscribe is a messaging pattern where publishers publishes messages to subscribers. The use of protocols based on pub/sub like Message Queuing Telemetry Transport (MQTT) is very essential when response time, lower battery, bandwidth and throughput usage are on the first place for future solutions. In this paper, a secure Elliptic Curve Cryptography (ECC) protocol using Publish / Subscribe lightweight protocol has been proposed for creating a secure tunnel between IoT devices and Cloud Computing, and that can allow a very fast communication also it's a light protocol that will not exhaust the resources of the IoT object. In fact, we use the AVISPA tool for a formal verification of our proposed protocol.

Keywords: *Security; Cloud Computing; Elliptic Curve Cryptography; Internet of Things; MQTT.*

1. INTRODUCTION

With a weak usage, IoT or Internet of things has become a common term in our society. By 2020, and thanks to the internet a hundred thousand objects will be connected to the Internet that communicate. On the other hand, the technology of Cloud Computing has become very usable. For the general public, Cloud computing is materialized in particular by digital data storage and sharing services such as Box, Drop box, Microsoft One drive or Apple I Cloud, where users can store personal content (photos, videos, music, documents). And access it anywhere in the world from any connected device. Most of the time, we talk about IoT and Cloud computing as two separate concepts. With an enormous amount of data generated by smart objects, where can we store them and how can we run them? The answer to this question is mixing the Cloud, and associating it with the IoT. Not to mention that there are a vast services that this integration can offer to humanity. Cloud-IoT is a new concept can join the wave of new technologies. However, the security continues to be the major issue while getting connected to Cloud for using its resources. Indeed due to the limited resources of connected objects such as memory size and processing capability, the use of cryptographic systems such as RSA will not be practicable to be implemented on IoT devices, and that of course is of because their security level that depend on the length of the keys. In addition to the fact that these schemes, being based on HTTP are complex enough to exhaust the resources of the object. Even if the server does the math and the distribution of the keys, we all know that HTTP consumes bandwidth. In fact the main characteristic of a smart device is its real time responding and interaction with its environment; If the problem of storage and processing has been solved by using the robust capacity of the Cloud. Then what is the purpose of having the IoT-Cloud solution if we use security

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

algorithms that exhaust the resources of the device.

In this paper, a secure ECC scheme for embedded devices and Cloud servers has been proposed. The proposed protocol ensures mutual authentication and secures communication between these objects and Clouds using the publish/subscribe pattern such as Message Queuing Telemetry Transport (MQTT). The rest of the paper is organized as follows. In Section 2 we define the Cloud-IoT concepts. In section 3 we will put the light on our motivation and contribution. In Section 4 we will discuss the preliminaries of elliptic curve cryptography. In Section 5, we will discuss related work and security issues in collaborating embedded devices with Cloud. In section 6, we will propose a novel ECC security scheme between the embedded device and Cloud server has been proposed. In section 7, a performance analysis has been done. We verify in section 8 the proposed protocol using Automated Validation of Internet Security Protocols and Applications (AVISPA) tool. Lastly, we will concludes the section in 9 concludes.

2. CLOUD-IOT PARADIGM

A connected object to the Internet, is an object with a certain level of intelligence that can communicate with others based on M2M communication. The birth of the IoT came only for one reason, is to meet our daily needs without the intervention of humans, but with an interaction with its environment by collecting an incalculable number of data, in order to build its own knowledge base. Unfortunately, these objects have an insufficient capacity in terms of storage, energy and robustness. If the data is collected and subsequently deleted due to storage inefficiency, why do we bring them together in the first place? In addition, Cloud computing is now mature and can offer storage capacity, robustness and verification. Not to mention services for the analysis and processing of data that can be of very great use of objects. An integration between the Cloud and IoT will be welcomed with open arms in order to create a homogeneous environment between the intelligibility of the objects and the robustness of the Cloud. For instant each researcher has a vision how this integration should be [1][2][3]. For us the hypothesis is, why not create a Cloud-IoT environment, offering on-demand services for each domain listed in the Internet of things sub-section. As we have already mentioned, the Cloud is not enough in terms of storage considering the immense demand of the IoT. Recently a new orientation appeared named Fog Computing. According to authors, the Fog is simply a Cloud that is close to the ground [4]. Its basic principle is conserving and treating data close to the place of collection; that is to say close to the sensor or the connected object. This of course, will allow us to significantly reduce the flow of data across the network. Despite this Fog will never fill in the functionality of Cloud computing [5]. It that is fair enough to say Cloud and Fog Computing complement each other.

3. MOTIVATION AND CONTRIBUTION

Due to the rapid evolution of the IoT-Cloud concept, hundreds of communication between the IoT devices and the Cloud will take place, which will transport the data flow, let us say for each user. The information sent to the Cloud via network channels is considered of great importance. Imagine that this data is not secure enough. With no doubt, this will generate a huge problem and bring about some serious consequences. Recently the research in IoT-Cloud paradigm has become very active, the development and implementation of platform in different sector. But rarely, where we found a community deals with the security aspect especially between the IoT objects and the cloud computing. For instance security researchers are leaning towards this problem, to secure communication between IoT devices and the Cloud server. There are a few security algorithms which addresses this problem and our goal in this article, is to propose a new secure scheme that can secure the communication between IoT device and the Cloud based on ECC and using the publish/subscribe pattern. The complexity of this mathematical problem is what makes the security methods very powerful, and the lightweight of the publish/subscribe pattern makes it very attractive for IoT devices. The benefits of this research is to put the light on this issue and to present a new security protocol which is the combination of ECC and pub/sub that is very powerful as it makes our proposed protocol different from others and very suitable for Cloud-IoT.

4. PRELIMINARIES

Before defining the elliptic curves [6], we must put the point on a very important notion, which is the cyclic group. Cyclic group, is a group whose elements are the multiples of a. It's about multiple classics (Z, +) or multiple power (Z, x). The element a is the generator. The order of the group is its number of elements. For example, if $G =$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$\{a^0, a^1, a^3, a^4\}$, next element is a^4 who will be the a^0.

4.1 Introduction to elliptic curve (EC)

An elliptic curve E defined on r is a smooth curve given by a Weierstrass equation:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

We will consider in what follows an elliptic curve, is a curve that is drawn by the points that will solve the following equation:

$$E = (x, y)|y^2 \equiv x^3 + ax + b \; with \; a, b \in K \quad (2)$$

a and b will have to fulfil the following condition$4a^3 + 27b^2 \neq 0$, K can be in the following fields $\{\mathbb{R}, \mathbb{Q}, \mathbb{C}, \mathbb{Z}/p\mathbb{Z}\}$.

Proposition:

Let E be an elliptic curve defined on a field K, and two points P, Q ∈E(K), L the line connecting P to Q (the tangent to E if P = Q) and R the third intersection point of L to E.Let L be the vertical line passing through R. We define P + Q ∈E(K) as the second point of intersection of L' with E. With the law of composition (E (K), +) is an abelian group whose neutral element is the point to infinity (O).

- **Point addition:** With 2 distinct points, P and Q, the addition is defined as the negation of the point resulting from the intersection of the curve, E, and the line defined by the points P and Q, giving the point, R.

$$P + Q = R \rightarrow (x_p, y_p) + (x_q, y_q) = (x_r, y_r)$$
$$x_r = \lambda^2 - (x_q + y_q)$$
$$y_r = \lambda \times (x_p - x_r) - y_p$$
$$with \; \lambda = \frac{(y_p - y_q)}{(x_p - x_q)}$$

- **Point doubling:** When the points P and Q are coincident, the addition is similar, except that there is no straight line defined by P and Q, so the operation is closed using the limit case, the tangent to the curve E, to P and Q. This is calculated as above but with a :

$$\lambda = \frac{(3x_p^2 + a)}{2y_p}$$

- **Vertical point:** The straight line joining any point P and its symmetrical relative to the horizontal axis, noted -P, is a vertical line, the third point of intersection with the curve is the point at infinity (which is its own symmetrical with respect to the abscissa axis), hence P + (-P) = 0.
- **Double-and-add:** The simplest method is the double-and-add method, similar to multiply-and-square in modular exponentiation. The algorithm works as follows: To compute DP, start with the binary representation for$d = d_0 + 2d_1 + 2^2d_2 + \cdots + 2^md_m \; with \; [d_0 \ldots d_m] \in$ 0,1.

4.2 Elliptic curve Cryptography (ECC)

4.2.1 What is ECC?

To get started, the RSA keys that have the recommended size, keep increasing to maintain sufficient encryption strength, from 1024 bits to 2048 bits a few years ago, are the most common used for SSL certificates. An alternative to RSA keys are the ECC keys. These two types of master keys share the same important property of being asymmetric algorithms (a key to encrypt and a key to decrypt). However, ECC can offer the same level of encryption power for much shorter keys, providing better security while reducing computing requirements.

4.2.2 What are the differences between RSA and ECC?

The key differentiation between the ECC and RSA is the size of the key compared to the encryption strength.

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Figure 1. Key comparison

4.2.3 Why use it?

The shorter keys make ECC a very attractive option for devices with storage or processing power is limited, which is becoming increasingly common in the era of the Internet of Things. For more traditional Web server use cases, shorter keys can be transcribed into faster SSL negotiations (which can lead to an acceleration of the loading speed of the web pages) and a reinforced security.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.2.4 How it works?

Example: Deffiehelman protocol

We will need to understand the notion of scalar multiplication. This group is needed to implement the DH protocol. P is a point that belongs to elliptic curve E.P is a point that belongs to elliptic curve E.

$$P \in E$$
$$K \in \mathbb{Z}$$
$$Q = KP \text{ with } Q \in E$$
$$Q = P + P + P + \cdots P \} K \text{ times}$$

So how do we use this property to create a cryptosystem based on elliptic curves? We need a one-way function. Is a function that can be easily calculated, but that is difficult to reverse - that is, given an image, it is difficult to find an antecedent.

ECDLP: Elliptic Curve Discrete Logarithm Problem

We suppose a curve $E(\mathbb{Z}/n\mathbb{Z})$. By giving a $Q,K \in E(\mathbb{Z}/n\mathbb{Z})$, with Q a multiple of P. We need to find K that solves the following equation Q = KP. It is a difficult problem to solve. This is called, the discrete logarithm problem or (ECDLP).

Another very important point to know is the point generator.

$G \in E(\mathbb{Z}/n\mathbb{Z})$, which generates a cyclic group.
Ord(G) = n, number of cyclic group element which gives KG = O.
Cofactor: $h = \frac{|E(\mathbb{Z}/n\mathbb{Z})|}{n}$, number of points in the curve the ideal is h=1

Let's summarize the parameters we need:

{P, a, b, G, n, h}
p: Field (modulo P)
a,b : Curve parameter E
G: Points generator
n: ORD(G)
h: Cofactor

5. RELATED WORKS

The security of data generated by the connected objects and transferred to the Cloud, requires significant resources such as storage capacity, processing and energy… Unfortunately, the security algorithms used nowadays to secure these objects. Either are vulnerable to attack, or require a huge time of calculation which will eventually exhaust the resources of the objects.

Recently, to reduce the computing time for smart device, schemes based on elliptic curve have been implemented. They chose the elliptic curve for many reasons, one of the main reasons is its key size which is very small compared to other asymmetric cryptosystems, as shown in Figure 1. And also its complexity; as its discrete logarithm is very difficult to calculate. In 2009 Yang and Chang [7], based on Tian et al's authentication [8], a scheme with mutual authentication and a session key agreement between the user and the server. The server is responsible for initializing the parameters and distributing the public key. This method is very interesting, it does not exhaust the resources of the device, since it is the server that does all the work, but unfortunately this algorithm suffers from the offline password guessing, and the clock synchronization [9]. In fact, it does not provide all the security necessary for an IoT device. In 2012 Hafizul et al. [10] by demonstrating the vulnerability of Debiao et al's scheme against some cryptographic attacks. He proposed a scheme consisting of four steps that we find interesting. Initialization phase, client registration, mutual authentication with key agreement and finally changing and updating the local private key phase. Unfortunately again this scheme suffers from the password guessing and does not hide the identity of the client. Other protocols based on ECC have been proposed for smart devices by Granjal et al. [11], Ray et al. [12] and Jiang et al. [13]. Another have been proposed for IoT using RFID systems also based on ECC, was proposed by Moosavi et al. [14].

Not long ago in 2015, a novel protocol appeared, proposed by Kalra and Sood [15], who have gained experience from other previously discussed algorithms. This scheme, is very interesting, it proposes a mutual authentication to secure the communication between IoT devices and the Cloud using HTTP cookies, for smart device that are HTTP clients. The use of cookies to develop a mutual authentication for smart devices, was very innovative. But in 2017 Kumari et al. [16] after the analysis their scheme, they showed that this algorithm is vulnerable against offline password guessing and insider attack. That is to say, this scheme does not provide device anonymity.

The algorithms that we have just mentioned, are based on HTTP and use the ECC. Despite the security vulnerabilities found and the consummation of the bandwidth that exhausts the resources of IoT devices, they provided a solid base for future research. In what follows we will quote some works based on Sub / pub and uses the ECC. Few works have thought of developing lightweight security schemes by combining sub / pub and ECC. The majority of researchers have focused on

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

studying sup / pub architecture network and its importance in an IoT-Cloud environment, but they have neglected a very important aspect which is security. In [17] a secure group communication was proposed, but the problem with such system is that group key management is not flexible for the sub / pub system. In [18] has analyzed group management for the secure distribution of pub/sub network events. Unfortunately, the number of keys has increased exponentially as subscribers.

6. PROPOSED PROTOCOL

In this section, we propose an ECC based authentication protocol for smart devices that are MQTT subscribers. We quoted in the previous section the known works to secure the communication between a connected object and the Cloud or fog computing. The idea of using MQTT for smart device authentication is novel. In this section, we propose a lightweight security mechanism based on MQTT. In an MQTT-based communication the broker is a very essential element for the rapid distribution of messages from a publisher to a subscriber according to a given topic. Our design is not very complicated, it is based on connecting several subscribers who are registered in a topic, according to the function or the task that this smart device occupies, to a Cloud or Fog computing via a broker who is in charge of dispatching messages. Once the Cloud or fog receives this message it takes decision from the information received, then publishes this decision to the subscribers concerned and so on. This decision-making must be secure because of the importance of information circulates, and it's here where we act. In our design we assume that the Broker has sufficient storage and calculation capacity to generate and distribute the keys to the sub for each session. We note that a publisher in our case the Cloud, must be a subscriber before having the ability to be a publisher. In fact the system enforces integrity and access control of messages under a given topic by employing authorization and encryption for publishers and decryption keys for subscribers.

6.1 System Security requirements

In order to make a secure system we need to respect some requirement including mutual authentication, confidentiality, anonymity and forward secrecy. In our protocol we discussed these requirements.

Mutual authentication: Authentication is a process that ensures and confirms user's identity. It is the most essential requirements first and foremost as the Subscribers and the Broker must authenticate each other for secure communication.

Confidentiality and integrity: The message sent from the subscriber to the publisher via the broker must be discreet and kept against the modification. Therefore, to ensure confidentiality, the Subscribers and Broker transmit encrypted information so that only they can recognize it.

Anonymity: Anonymity means that intruders cannot trace the subscriber's information in place of a legitimate Broker.

Forward secrecy: If the information transmitted from a subscriber is intercepted that constitutes a serious privacy issue. That's why the information transmitted from the subscriber should not be traced.

6.2 Security procedures

In this section, we will detail the proposed security mechanism. Our mechanism consists of 3 parts. The first is the initialization where the broker initiates the parameters necessary for the generation and distribution of keys. The second is the subscription between the objects and the broker is also composed of sub-section that we will detail later. And finally the third is the publication, it's where the publisher (Cloud) publishes information to subscribers via the Broker. Table 1 denotes the notations used in proposed protocol.

TABLE 1: NOTIONS USED IN THIS PAPER

Notation	Description
ID_i	Identity of device i
ID_{Bro}	Identity of broker
$Topic_i$	Subscription topic
K_m	Random master key
K_i	Key for device i
R	Random number
Z_p	Finite field group
P	Large prime number of the order $>2^{160}$
N_1, N_2	Random numbers generated for ECC parameters
G	Generator point of a large order n
h(.)	Cryptographic one-way hash function
K	Mutual auth key
M	Message
C	Encrypted message
$\oplus$	XOR operator
$\|$	Concatenation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6.2.1 Initialization

As we have already mentioned, in our design it is the Broker who is in charge of the generation and distribution of keys. But before this step, the Broker must initiates its parameters by first choosing an elliptic curve equation$y^2 = x^3$+ax+b over Z_p, where $Z_p(P > 2^{160})$is the finite field group. The Broker selects two field elementsa, b $\in$ Z_p. a and b must satisfy the following condition $4a^3 + 27b^2 \neq 0$. G is the base point of the elliptic curve with a prime order$n > 2^{160}$, and O be the point at infinity such that n x G = 0. Then the Brokerchooses random master secret key K_m from Z_p and computes public key $L = G.K_m$. Finally broadcast the public parameters $K_p \leftarrow (E_p, G, p, L)$.

6.2.2 Subscription

In this step a mutual authentication will be made between the subscribers and the Broker. To do this two necessary phase will be developed. The first, is the registration phase in order to register the subscribers with the Broker. The second is the login & authentication phase, in this phase the Broker makes sure of subscribers information then a mutual authentication will be made.

a. Registration phase:

In order to register with the Broker, the Subscriber requests by presenting

$I_i \rightarrow h(ID_i \| Topic_i \| rG)$.

Step 1: $Subscriber_i \rightarrow Broker : I_i$, *The Broker generates* K_i

The Broker generates for every Subscriber $A_i \rightarrow h(K_m \| K_i)$, where $K_i = h(N \| I_i \| ID_{bro}) \oplus topic_i$, and stores $A_i' = A_i.G$ on the Subscriber. The Broker also calculates the security parameters $K_s = K_m \oplus K_i$, $B_i = h(K_s \oplus I_i \oplus A_i')$ and stores $B_i' = B_i.G$, K_i and $Topic_i$ corresponding to the identity ID_i of the subscriber i in its database.

Step 2: $Broker \rightarrow Subscriber_i : K_i, A_i'$

b. Login & Authentication phase :

Before every login, the Subscriber selects a random number N_1 and calculates an ECC point $P_1 = N_1.G$ and stores it in its memory.

Step 1: *The subscriber Calculates ECC point* P_1

In order to login with the Broker, the device calculates the ECC point $P_2 = h(N1.A_i')$.

Step 2: *The Subscriber sends* P_1, P_2. *and the* K_i *to the Broker.*

After receiving the parameters on login request, the Broker calculates $A_i \rightarrow h(K_m \| K_i)$ by calculating the $K_m = K_s \oplus K_i$. It then calculates the point $P_2^* = h(P_1.A_i)$. The Broker then checks whether the value of P_2^* is equal to the received value of P_2.

Step 3: *Broker checks* $P_2^* ? = P_2$

Then the Broker selects a random number N_2 and calculates the ECC point $P_3 = N_2.G$, $P_4 = N_2.B_i'$. And Sends P_3, P_4, and K_s to the Subscriber.

Step 4: *Broker* $\rightarrow$ *Subscriber:* $P_3, P_4 and K_s$

The subscriber then calculates $B_i = h(k_s \oplus I_i \oplus A_i')$ and calculate ECC point $P_4^* = P3.B_i$, and compares the value of P^*4 with the received value of P_4.

Step 5: *The subscriber checks* $P_4^* ? = P_4$

Then, the device calculates $V_i = h(P_2 \| K)$, where $K = N_1.P_3$ and sends V_i to the broker. The broker calculates $V_i^* = h((P_1.A_i) \| K^*)$ where $K^* = N_2.P_1$. And compares the value to the received value of V_i to authenticate the device.

Step 6: *The Broker checks* $V_i^* ? = V_i$

After mutual authentication between the subscriber and the Broker.

6.2.3 Publication

After a mutual authentication. The publishing phase came, where publisher send a cipher message to subscribers via the Broker.

Step 1: *Publisher* $\rightarrow$ *Broker:* ID_{pub}.

Publisher requests publication by $ID_{pub} = h(topic_i \| ID_i \| G.K)$.

Step 2: *Checks the identity of the Publisher.*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Computes $ID^*_{pub} = h(topic^*_i \| ID^*_i \| G.K^*)$, and compares it with ID_{pub}. If it's true then the Broker sends $\{ ID_{bro} \| ID_{pub} \}$

Step 3: *Publisher* $\rightarrow$ *Broker: Pub*

The publisher computes $C_{pub} = r.G.K_i \oplus M$ and sends to The Broker $Pub = h(Topic_i \| ID_i \| rG \| C_{pub})$.

Step 4: *Broker* $\rightarrow$ *Subscribers:* C_i. The Broker computes $C_b = r.G.K_s \oplus C_{pub} = rGK_s \oplus rG.K_i \oplus M$, $C_p = K_m \oplus M$. Then Computes $C_i = C_b \oplus rGK_s = K_m \oplus M \oplus rGK_s$, $C_i = rGK_i \oplus M$. Finally The Broker sends $\{ C_i \}$ to Subscribers registered in $Topic_i$.

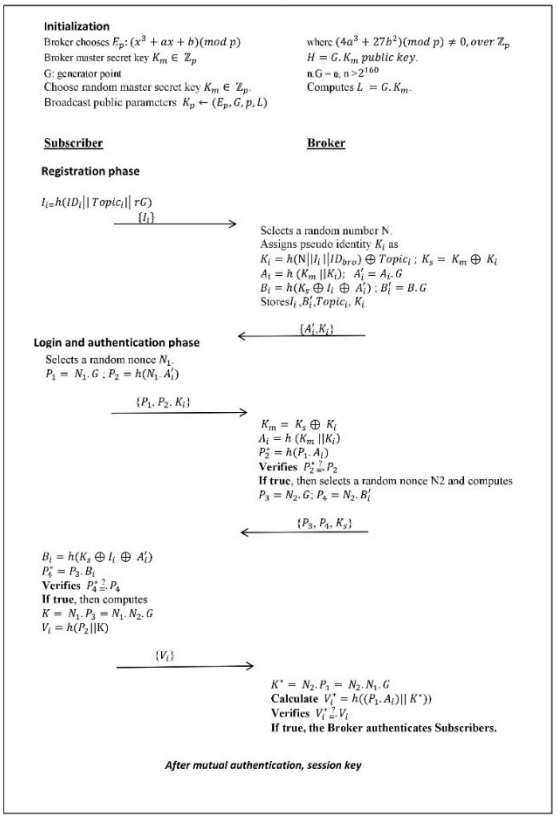

Figure 2. Summary of the Subscribing phase

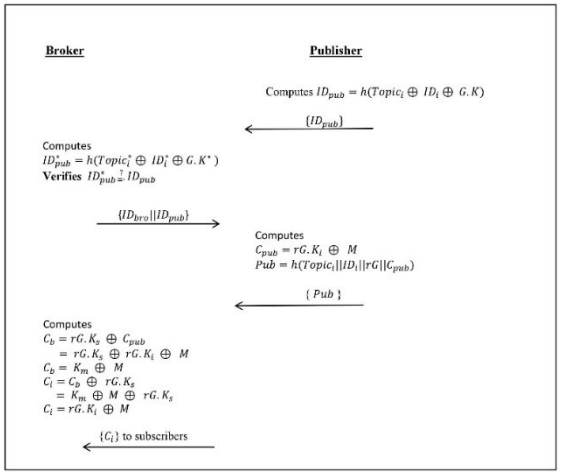

Figure 3. Summary of the Publishing phase

7. PERFORMANCE ANALYSIS

Our protocol is composed of subscription, publication and the broker who is in charge of dispatching information. The proposed protocol could be evaluated in terms of various overhead, security parameters and scalability. Communication between publishers and subscribers is usually secured with SSL / TSL. In This section, we will evaluate and compare our protocol with SSL/TSL.

SSL/TSL has 13 handshakes, is the most widely used TSL handshake version. As the message size in each handshake varies, from Table 2 we will try to approximate SSL/TSL message size in each handshake connection. We should point out that TSL Record header for each record (5 bytes), as well as TSL Handshake header (4 bytes). We have 4 Records exchanged (20 bytes). Each message has the handshake header, so we have 7 times the Handshake (28 Bytes).The total overhead of a single new SSL/TSL connection comes to about 1789 Bytes. (20 + 28 + 170 + 75 + 1500 + 130 + 2 + 24 = 1789 bytes).The total overhead of an existing SSL/TSL connection comes to about 332 Bytes. (15 +16 + 202 + 75 + 2 + 24 = 332bytes).

TABLE 2: MESSAGE SIZES OF SSL/TSL

Message type	Message size
ClientHello	160–170 bytes
Server Hello	70–75 bytes
Certificate	800–1500 bytes
ClientKeyExchange	130 bytes
ChangeCipherSpec	1 byte
Finished	12–36 bytes

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7.1 Communication Cost

While calculating the cost of the protocol, we assumed that the length of the random number (N, N_1, N_2) is 160 bits, identity is 160 bits, pseudo-identity K_i is 160 bits and the security parameter K_s, V_i .is 160 bits. Also, the output of one way hash function (SHA-1) is 160 bits. In proposed protocol, we consider the elliptic curve cryptosystem (ECC) with 160 bit. An ECC point P = (x_p, y_p) needs (160 + 160) = 320 bits. So communication cost during a subscription for:

- Subscribes/publishers: [P_1, P_2, K_i, V_i] = [2*360 + 2*160] = 960bit = 120 Bytes.
- Broker: [P_3, P_4, K_s] = [2*320 + 160] = 800 bit= 100 Bytes.

Communication cost during a publication for:

- Subscribes/publishers: [ID_{pub} , PUB] = [2*160] = 320 bit= 40Bytes.
- Broker: [$ID_{pub}, IDBro, C_i$] = [3*160] = 480bit= 60Bytes.

7.2 Storage cost

We will compute the memory needed by the devices and the broker to store their security parameters.

- Subscribes/publishers: [$A_i^{'}, K_i$] = [320+160] = 480bit = 60 Bytes.
- Broker: [$I_i, B_i^{'}, Topic_i, K_i$] = [4*160] = 640 bit= 80Bytes.

7.3 Comparison

TABLE 3: PERFORMANCE COMPARISON BETWEEN OUR SCHEMES AND SSL/TLS

Parameter type		Our Scheme		SSL/TSL	
		Publisher/ subscriber	Broker	Publisher/ subscriber	Broker
Subscription	Computational overhead	2 XOR+ 4 T_{ecm}	6 XOR + 4 T_{ecm}	-	-
	Storage overhead	60 bytes	80 bytes	−/Over 235 bytes	Over 1554 bytes
	Communication overhead	120 bytes	100 bytes	−/Over 235 bytes	Over 1554 bytes
Publication	Computational overhead	3 XOR	4 XOR	-	-
	Communication overhead	40 bytes	60 bytes	−/Over 235 bytes	Over 1554 bytes

T_{ecm} The computational cost of a ECC point multiplication operation

8. FORMAL VERIFICATION OF THE PROTOCOL USING AVISPA

Avispa (automated validation of internet security protocols and applications) is a push-button tool for the automated validation of internet security-sensitive protocols and applications. It provides a modular and expressive formal language for specifying protocols and their security properties, and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques [19]. Avispa provides a language called the high level protocol specification language (hlpsl) [20] for describing security protocols and specifying their intended security properties, as well as a set of tools to formally validate them. An hlpsl specification is translated into the intermediate format (if), using a translator called hlpsl2if. If is a lower-level language than hlpsl and is read directly by the back-ends to the avispa tool. Note that this intermediate translation step is transparent to the user, as the translator is called automatically. The interested reader can find more about the if in the avispa user manual and in the avispa deliverable which discusses if [19]. The if specification of a protocol is then input to the back-ends of the avispa tool to analyze if the security goals are satisfied or violated. Protocol specification in hlpsl is divided into roles that describe the actions of one single agent in a run of a protocol or sub-protocol. Also, hlpsl code must be open to automated formal analysis. The back-ends are implemented using formal methods and theoretical axioms and are used to provide protocol falsification, bounded and unbounded verification. Two backs ends used for analysis of the proposed protocol are on the fly model checker (ofmc) and cl-based attack searcher (cl-atse).

OFMC: This back-end builds the infinite tree defined by the protocol analysis problem and executes different symbolic techniques to search the state space in a demand-driven

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

way, i.e., on-the-fly. OFMC helps to detect attacks and verify the correctness of the protocol for a bounded number of sessions but without limiting the number of messages that an intruder can generate [21].

CL-AtSe: This back-end is used to detect the attacks on the protocol by using a set of constraints that are obtained by translating the security protocol specification written in Intermediate Format (IF).Detection of attacks and translation of protocol specifications that are designed based on the adversary's knowledge, are fully automated and are internally performed by the CL-AtSe model checker [21].

8.1 Specification of the protocol

We have implemented two basic roles for the *Subscriber* and the *Broker* in HLPSL related to the registration phase and login and authentication phase of our Protocol. Besides these two roles, the roles for the session, goal and environment in HLPSL has been implemented. The role of the initiator, *Subscriber* in HLPSL is given in Fig. 4. *Subscriber* first receives the start signal, updates its state from 0 to 1, which is maintained by the variable State, and then sends the registration request {Ii} securely to the *Broker* during the registration phase with the help of SB( ) channel. The type declaration channel *(*d*y)* declares that the channel is for the Dolev–Yao threat model [22], which indicates that the intruder (i) has the ability to intercept, analyze, and/or modify messages transmitted over an insecure public channel. After that Subscriber receives the message { } from Broker securely by the help of RB( ) channel. In this role, the played by X declaration indicates that the agent named in variable X plays in the role. By the declaration secret(Idi, R.G, ,sec1, Sub), it indicates that the information Idi , R.G and are only known to Subscriber . During the login and authentication phase, Subscriber sends the message { } to Broker through open channel. Subscriber then receives the message { P_3, P_4, K_s } from *Broker* via public channel. Finally, *Subscriber* replies with the message { V_i } to *Broker via* open channel. A knowledge declaration (generally in the top-level *Environment* role) is used to specify the intruder's initial knowledge. Immediate reaction transitions are of the form $X = | > Y$, which relate an event X and an action Y. By the declaration *witness (Sub, Bro, alice_bob_r1,* N_1*)*, we mean that *Sub* has generated the random nonce *n*1 freshly for the *Broker*. On the other hand, by the declaration *request(Bro, Sub, bob_alice_r2,* N_2 *')*, it indicates that subscriber's acceptance of the random nonce *n*2 generated for *Subscriber by Broker*. In a similar way, we have implemented the role for the *Broker* during the registration as well as login and authentication phases in Fig. 5.

Finally, the roles for the session, goal and environment of our proposed scheme are provided in Fig. 6. All the basic roles: Subscriber and Broker are the instances with concrete arguments in the role of the session. The top-level role (environment) is always defined in HLPSL specification. In HLPSL, the intruder (i) also participates in the execution of protocol as a concrete session as shown in Fig. 6. In our implementation, there are two secrecy goals and two authentication goals.

8.2 Protocol Specification in AVISPA

The analysis of the proposed scheme is performed by defining the protocol in HLPSL and testing it using AVISPA back-ends. This type of analysis is useful in locating design flaws and problems that would be very difficult and expensive to solve once the protocol has been deployed in real systems. In the verification phase via Avispa. We concentrate on checking the security level of subscription phase between subscribers and the broker. There are two basic roles "subscriber" and "Broker". Here, we represent the HLPSL coding of "subscriber" role in Fig.4 and "Broker" role in Fig. 5.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
role subscriber (Sub, Bro: agent,
                 SKedics: symmetric_key,
                 H : hash_func,
                 SB, RB: channel(dy))

played_by Sub
def=
local State: nat,
    Idi,Topic, Ii, Ki, Idbro, Ks, Km: text,
    G,R, N1, P1, P2, N2,N, SK, Vi: text,
    F : hash_func
const sec1, sec2, alice_bob_n1, bob_alice_n2 : protocol_id
init State := 0
transition
% Registration phase
1. State = 0 /\ RB(start) =|>
        State' := 1 /\ R' := new()
                  /\ Ii' := H(Idi.F(R'.G).Topic)
                  /\ secret({Idi,F(R'.G),Topic}, sec1, Sub)
% Send the registration request {Ii} to broker securely
% via secure channel
                  /\ SB({Ii'}_SKedics)
% Receive resgistration reply {PIdi, Ck'} from broker securely
2. State = 1 /\ RB({xor(H(N'.Idbro.H(Idi.F(R'.G).Topic)),
Topic).F(H(Km'.xor(H(N'.Idbro.H(Idi.F(R'.G).Topic)),Topic)).G)}_SKedics) =|>
        State' := 3 /\ secret({Idbro}, sec2, Sub)
% Login and authentication phase
        /\ N1' := new() /\ P1' := F(N1'.G)
        /\ P2' := H(N1'.F(H(Km'.xor(H(N'.Idbro.H(Idi.F(R'.G).Topic)),Topic)).G))
% Send login request {P1, P2, Ki} to broker via open channel
        /\ SB(P1'.P2'.xor(H(N'.Idbro.H(Idi.F(R'.G).Topic)), Topic))
        /\ witness(Sub, Bro, alice_bob_n1, N1')
% Receive message {P3, P4, Ks} from CS via open channel
3. State = 3 /\ RB( F(N2'.G).
F(N2'.F(H(xor(xor(xor(Km',xor(H(N'.Idbro.H(Idi.F(R'.G).Topic)),
Topic)),H(Idi.F(R'.G).Topic)),F(H(Km'.xor(H(N'.Idbro.H(Idi.F(R'.G).Topic)),Topic)).G))).G))
.xor(Km',xor(H(N'.Idbro.H(Idi.F(R'.G).Topic)), Topic)) ) =|>
        State' := 5 /\ SK' := F(N1.F(N2'.G))
                  /\ Vi' := H(F(N1.F(H(Km'.Ki).G)).SK')
% Send message {Vi} to broker via open channel
                  /\ SB(Vi')
% subscriber's acceptance of random nonce n2 generated for sub by broker
                  /\ request(Bro, Sub, bob_alice_n2, N2')
end role
```

Figure 4. Subscriber Role

```
role broker (Sub, Bro: agent,
                 SKedics: symmetric_key,
                 H : hash_func,
                 SB, RB: channel(dy))
played_by Bro
def=
local State: nat,
    Idi, Idbro, Km, Ai, Aii : text,
    Ki,R, G, N1, N2,N, P3, P4,Topic : text,
    Bi, Bi1, Ks : text,
    F : hash_func
const sec1, sec2, alice_bob_r1, bob_alice_r2 : protocol_id
init State := 0
transition
% Registration phase
% Receive registration request <Ii> from subscriber securely
% subscriiber securely
1. State = 0 /\ RB({H(Idi.F(R.G).Topic)}_SKedics) =|>
        State' := 2 /\ N' := new() /\ secret({Idi,F(R.G)}, sec1, Sub)
                  /\ Km' := new()
                  /\ Ki' := xor(H(N'.Idbro.H(Idi.F(R.G).Topic)), Topic)
                  /\ Ai' := H(Km'.Ki')
                  /\ Aii' := F(Ai'.G)
                  /\ secret({Idbro}, sec2, Sub)
% Send resgistration reply {PIdi, A'} to subscriber securely
                  /\ SB({Ki'.Aii'}_SKedics)
% Login and authentication phase
% Receive login request {P1, P2, Ki} from subscriber via open channel
2. State = 2 /\
RB(F(N1'.G).F(N1'.F(H(Km'.xor(H(N'.Idbro.H(Idi.F(R.G).Topic)),Topic)).G)).xor(H(N'.Idbr
o.H(Idi.F(R.G).Topic)), Topic)) =|>
        State' := 4 /\ N2' := new() /\ P3' := F(N2'.G)
                  /\ Bi' := H(xor(xor(xor(Km',xor(H(N'.Idbro.H(Idi.F(R.G).Topic)),
Topic)),H(Idi.F(R.G).Topic)),F(H(Km'.xor(H(N'.Idbro.H(Idi.F(R.G).Topic)), Topic)).G)))
                  /\ Bi1' := F(Bi'.G)
                  /\ P4' := F(N2'.Bi1')
                  /\ Ks' := xor(Km',xor(H(N'.Idbro.H(Idi.F(R.G).Topic)), Topic))
% Send message {P3, P4, Ks} to subscriber via open channel
                  /\ SB(P3'.P4'.Ks')
                  /\ witness(Bro, Sub, bob_alice_r2, N2')
% Receive message {Vi} from subscriber via open channel
3. State = 4 /\ RB(H(F(N1'.F(H(Km'.Ki).G)).F(N1'.F(N2'.G)))) =|>
        State' := 6 /\ request(Sub, Bro, alice_bob_r1, N1')
end role
```

Figure 5. Broker Role

After mutual verification is accomplished, a session is setup between subscriber and broker. In order to define the session of the protocol, composed roles are defined after defining the basic roles. Fig. 6depicts the role "session". In session segment, both the basic roles (subscriber, broker) are instanced with concrete arguments. "Session" role contains global constants and a composition of other roles, where the intruder may act as the role of a legitimate user. Then a top-level role "Environment" is defined that is depicted in Fig. 6The constant "i" is used to personify as an intruder. The intruder also participates in the execution of the protocol as a concrete session. The properties specified in "Goal" section are also depicted in Fig. 6.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
role session (Sub, Bro: agent,
SKedics: symmetric_key,
H: hash_func)
def=
local SB1, RB1, SB2, RB2: channel(dy)
composition
subscriber (Sub, Bro, SKedics, H, SB1, RB1)
/\ broker (Sub, Bro, SKedics, H, SB2, RB2)
end role

role environment()
def=
const sub, bro: agent,
skedics: symmetric_key,
h, f: hash_func,
p1, p2, ki, p3, p4, ks, vi: text,
sec1, sec2, alice_bob_r1, bob_alice_r2: protocol_id
intruder_knowledge = {sub, bro, h, f,
p1, p2, ki, p3, p4, ks, vi}
composition
session(sub, bro, skedics, h)
/\ session (i, bro, skedics, h)
/\ session (sub, i, skedics, h)
end role

goal
secrecy_of sec1
secrecy_of sec2
authentication_on alice_bob_r1
authentication_on bob_alice_r2
end goal
environment()
```

Figure 6. Composed role

8.3 Formal Security Analysis of the Protocol

The proposed scheme is analyzed using OFMC and CL-AtSe back-ends and the results are shown in Fig. 7 and Fig. 8. Under this model, the intruder has full control over the network in such a manner that all messages sent by the agents corresponding to the roles are available to the intruder. The intruder may intercept, analyze and, or modify messages as long as he knows the required keys. He can act as any agent and send any message to some other agent participating over the communication channel.

The simulation results show that the proposed protocol is secure

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/Test_algo_these.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed  : 0 states
 Reachable  : 0 states
 Translation: 0.01 seconds
 Computation: 0.00 seconds
```

Figure 7. Output of ATSE backend

```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/Test_algo_these.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.02s
 visitedNodes: 4 nodes
 depth: 2 plies
```

Figure 8. Output of OFMC back

9. CONCLUSION

The convergence between IoT and Cloud clearly has advantages in several fields, transportation and logistics domain, healthcare domain, smart environment domain as well as personal and social domain. In this paper, we defined the IoT-Cloud concept, we have shown the utility and importance of using the elliptic curve compared to the RSA, and we proposed a lightweight security mechanism based on MQTT and the complexity of the ECC. The idea of using MQTT for smart device authentication is novel and who satisfies all the essential security requirements needed for this convergence. A comparison of the performance of our scheme with the TLS/SSL in section 7 was done, and the result was very satisfying. A formal security analysis based on attack model was done and that proves that the protocol is robust against all the security threats. Finally we verified our protocol using the automated verification of the protocol using AVISPA tool. Results have shown that the protocol is safe. As part of future study, we need to implement the protocol on real IoT platform.

REFRENCES:

[1] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescap. On the integration of Cloud computing and internet of things. In Future Internet of Things and Cloud (FiCloud), 2014 International Conference on, pages 23-30. IEEE, 2014.

[2] Stefan Nastic, Sanjin Sehic, Duc-Hung Le, Hong- Linh Truong, and Schahram Dustdar. Provisioning Software-De_ned IoT Cloud Systems. Pages 288-295. IEEE, August 2014.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[3] Mohammad Aazam, Imran Khan, Aymen Abdullah Alsa_ar, and Eui-Nam Huh. Cloud of Things: Integrating Internet of Things and Cloud computing and the issues involved. In Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on, pages 414{419. IEEE, 2014.

[4] Arslan Munir, Prasanna Kansakar, and Samee U. Khan. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. IEEE Consumer Electronics Magazine, 6(3):74-82, July 2017.

[5] Manuel Daz, Cristian Martn, and Bartolom Rubio. State-of-the-art, challenges, and open issues in the integration of Internet of things and Cloud computing. Journal of Network and Computer Applications, 67:99-117, May 2016.

[6] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. CRC press, 1996.

[7] Jen-Ho Yang and Chin-Chen Chang. An ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem. Computers & Security, 28(3- 4):138-143, May 2009.

[8] Xiaojian Tian, D.S. Wong, and R.W. Zhu. Analysis and improvement of an authenticated key exchange protocol for sensor networks. IEEE Communications Letters, 9(11):970-972, November 2005.

[9] Ding Wang, Ying Mei, Chunguang Ma, and Zhen-shan Cui. Comments on an Advanced Dynamic ID-Based Authentication Scheme for Cloud Computing. In WISM, pages 246-253. Springer, 2012.

[10] Sk Ha_zul Islam and G. P. Biswas. An improved ID-based client authentication with key agreement scheme on ECC for mobile client-server environments. Theoretical and Applied Informatics, 24(4), January 2012.

[11] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. End-to-end transport-layer security for Internet-integrated sensing applications with mutual and delegated ECC public-key authentication. In IFIP Networking Conference, 2013, pages 1-9. IEEE, 2013.

[12] Sangram Ray and G P. Biswas. Establishment of ECC-based Initial Secrecy Usable for IKE Implementation. July 2012.

[13] Rong Jiang, Chengzhe Lai, Jun Luo, Xiaoping Wang, and Hong Wang. EAP-Based Group Authentication and Key Agreement Protocol for Machine-Type Communications. International Journal of Distributed Sensor Networks, 9(11):304601, Novembre 2013.

[14] Sanaz Rahimi Moosavi, Ethiopia Nigussie, Seppo Virtanen, and Jouni Isoaho. An Elliptic Curve-based Mutual Authentication Scheme for RFID Implant Systems. Procedia Computer Science, 32:198-206, 2014.

[15] Sheetal Kalra and Sandeep K. Sood. Secure authentication scheme for IoT and Cloud servers. Pervasive and Mobile Computing, 24:210-223, December 2015.

[16] Saru Kumari, Marimuthu Karuppiah, Ashok Kumar Das, Xiong Li, Fan Wu, and Neeraj Kumar. A secure authentication scheme based on elliptic curve cryptography for IoT and Cloud servers. The Journal of Supercomputing, April 2017.

[17] Pawani Porambage, An Braeken, Corinna Schmitt, Andrei Gurtov, Mika Ylianttila, and Burkhard Stiller. Group key establishment for enabling secure multicast communication in wireless sensor networks deployed for IoT applications. 3:1503-1511.

[18] Chenxi Wang, A. Carzaniga, D. Evans, and A.L. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. Pages 3940-3947. IEEE Comput. Soc.

[19] AVISPA Team. AVISPA v1.1 User Manual. 2006.

[20] AVISPA Team. The High Level Protocol Speci_cation Language. 2003.

[21] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P. C. Hem, O. Kouchnarenko, J. Mantovani, S. Mdersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigan, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In Kousha Etessami and Sriram K. Rajamani, editors, Computer Aided Veri_cation, volume 3576, pages 281-285. Springer Berlin Heidelberg.

[22] DolevD,YaoAC (1983) On the security of public key protocols. IEEE Trans Inf Theory 29(2):198–208