A Robust Authentication Scheme for Observing Resources in the Internet of Things Environment

Mian Ahmad Jan, Priyadarsi Nanda, Xiangjian He, Zhiyuan Tan
Centre for Innovation in IT Services and Applications (iNEXT)
University of Technology, Sydney
Sydney, Australia

Ren Ping Liu
Computational Informatics, CSIRO
Sydney, Australia

***Abstract*—The Internet of Things is a vision that broadens the scope of the internet by incorporating physical objects to identify themselves to the participating entities. This innovative concept enables a physical device to represent itself in the digital world. There are a lot of speculations and future forecasts about the Internet of Things devices. However, most of them are vendor specific and lack a unified standard, which renders their seamless integration and interoperable operations. Another major concern is the lack of security features in these devices and their corresponding products. Most of them are resource-starved and unable to support computationally complex and resource consuming secure algorithms. In this paper, we have proposed a lightweight mutual authentication scheme which validates the identities of the participating devices before engaging them in communication for the resource observation. Our scheme incurs less connection overhead and provides a robust defence solution to combat various types of attacks.**

***Index Terms*—Internet of Things (IoT), CoAP, Authentication, Conditional Option, Resource Observation**

I. Introduction

Advances in the wired, wireless, cellular and sensor networks have laid a solid foundation for the Internet of Things (IoT). It is a novel paradigm which encompasses everyday physical world objects by enabling interaction among them via unique addressing schemes [1]. It is estimated that around 50 billion such objects will be connected to the internet by 2020[1]. Sensors, RFID tags, smart thermostats, PDAs, smart phones, gadgets etc. are some to mention in this context. These devices will be empowered to sense, process and control the physical world events and numerous phenomena of interest. Eventually, the IoT will lead us to the Internet of Everything (IoE), where people, data, objects and processes will be an integral part of it. We are moving to an era where the IoT-based internet will become ubiquitous by integrating the virtual world of information with the physical world of objects. This integration with the physical world will facilitate humanity in the long run. The refrigerators will automatically order groceries from the supermarkets which will be delivered upon payment verification.

To realize the above vision, the industry needs to adhere to a unified standard. Currently, each application has its own specifications and underlying software and hardware platforms. The devices require a scalable application layer for the interoperable communication. Also, a common programming model is required which will enable the developers to focus only on the application development rather than the hassle of worrying about the underlying platform [2]. The IoT comprises energy-starved sensor nodes at its core which are equipped with relatively small amount of memory; hence, the application code needs to run on the cloud and only the firmware and the network stack need to be nested at the core of each embedded device. Running applications on the cloud will serve two major purposes: the availability of ample memory space on the nodes and developing applications irrespective of the underlying hardware architecture.

Integrating everyday objects in the internet is not a straightforward process. Security is one obvious challenging task faced by the IoT as each object has its own features and requirements. First, the identity of each person, object and system connected with the internet needs to be established. In the absence of identity the intruders will gain access to the network and conduct security breaches. The consequences of these breaches vary from one application to another. These security threats are diverse in nature which ranges from disabling home security system to conveying false health readings to the practitioners and activating false fire alarms are some to mention in this context.

Despite all these security threats, most of the IoT products available in the market lack secure identities. As a result, we are about to use products which are vulnerable to various types of security breaches. These products will lead us to a new era of cybercrimes rather than improving our lives. As a result, the IoT will more likely become the Internet of Vulnerabilities (IoV). Recently, Proofpoint Inc.[2] a leading security firm uncovered cyber attack involving physical objects. This is considered as the first major security breach in the world of IoT. Over a period of less than two weeks, $750,000$ malicious emails were transmitted from more than $100,000$ devices. Interestingly, more than 25 percent of those devices were the physical objects including television, refrigerator and other house hold appliances. It was by far the simplest of attacks as misconfiguration and using default passwords were sufficient to compromise these physical objects. Hence, the IoT faces diverse challenges of thing-bots along with the traditional botnets.

In this paper, we have proposed a lightweight secure authentication algorithm which verifies the identities of the clients and servers participating in the network. We have chosen Constrained Application Protocol (CoAP) as the underlying application layer protocol for enabling communication among various physical objects.

This paper serves two major purposes. First, it authenticates the identities of the clients and server communicating with each other. Second, it provides various resources to the clients based on certain conditions specified in the request. Both these objectives are necessary for a robust and efficient communication system. The IoT acts as a hub for the heterogeneous devices having specific requirements in terms of data rate, latency, bandwidth, security and other performance metrics. The participating devices can easily be compromised to act as potential intruders in order to manipulate the network resources in an abnormal fashion. Hence, the identity of each device either as client or server needs to be verified in an energy-efficient manner. Upon authentication, actual data transmission commences among the end points. Each

[1]http://www.cisco.com/web/solutions/trends/iot/indepth.html

[2]http://www.proofpoint.com/about-us/press-releases/01162014.php

client specifies certain conditions based on its application requirement for resource observation. The conditional specific data transmission reduces the number of transmitted packets which has a direct impact on energy consumption, computation and bandwidth utilization of the communicating endpoints.

The paper is organized into six sections. In Section II, a brief overview of the problem statement is provided. In Section III, we first present our lightweight mutual authentication algorithm followed by the conditional resource observation in an IoT domain. In Section IV, we have provided the preliminary evaluation results based on our authentication algorithm. The related work on resource observation and authentication is presented in section V. Finally, the paper is concluded and the future research directions are provided in section VI.

II. Problem Statement

The Internet of Things incorporates devices from a very diverse background. These devices differ from each other in terms of their size, storage, energy consumption, computation, data rate and other performance metrics. Seamless and interoperable communication among the devices is enabled via sensors and actuators embedded in them. These miniature sensors give a unique ID to each participating device in an IoT paradigm. Sensors broaden the scope and scalability of today's internet by integrating them to the physical objects. However, it requires an effort on the application developers side because sensors are tiny, energy-starved and constrained on computation and storage capacity. Hence, there is a trade-off between the scalability and network resources.

Trust, privacy and security provisioning are challenging tasks in any communication network. However in the IoT, designing secure solutions are more difficult and complex due to the peculiar nature of the devices. Also, sensors are deployed at the core of the communication system, which makes it more difficult to design computationally complex but secure and robust algorithms. In the internet of today, various types of attacks and their defence mechanisms are well studied. However in the IoT domain, threats posed by various objects are unknown until they are deployed in the network. Hence, their dimension, scope and nature are yet to be observed.

Figure 1 depicts a small scale IoT environment, which is vulnerable to different types of attacks. Here, an intruder poses threats to various types of devices and information at a given time. Internet, sensor network and smart phone are susceptible to this attack. Any data coming from the attached devices will also be affected. Hence, a single intruder is capable to conduct a large scale attack. The intruder might intercept the sensor data, manipulate it and replays the malicious data in other parts of the network. Also, it might inject fabricated data of its own. In this figure, various types of objects are connected with the internet along with personal computers. Hence, large amount of data is at risk which might result in malfunctioning of the whole network. Off course, the severity of this attack depends on the intruder's battery-power, storage, computation and other features.

Like any other legitimate node, a malicious node also requires an ID to participate in an IoT communication. Each participating device needs to be validated and authenticated in order to establish its true identity in the network. In absence of ID validation and authentication, an attacker will always be capable of conducting a wide range of malicious activities. An intruder might establish multiple connections with the gateway node of the sensor network as shown in Figure 1. This is by far the simplest of the attacks where multiple network resources will be seized by the malicious device. This results in denial of services to the legitimate sensor nodes and ultimately causes

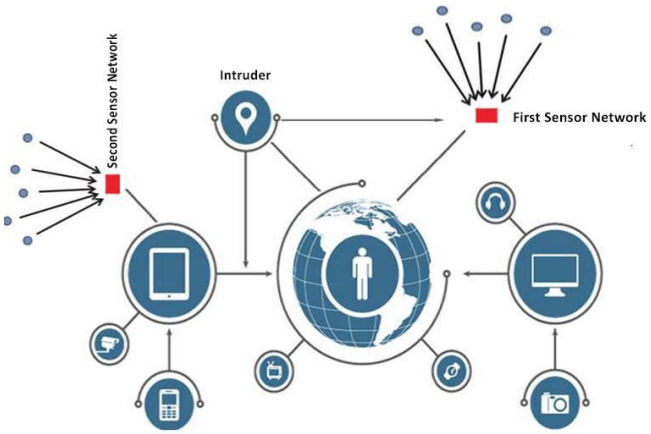

Fig. 1. A Vulnerable IoT Architecture

scarcity of resources in the network [3]. Wireless medium is shared among the network devices. The intruder might block access to the network resources by continuously emitting signals to interfere with the legitimate transmissions. This act is known as jamming [4] and its consequences are severe in an IoT environment as compared to wired and wireless networks because, physical objects differ significantly from each other in terms of various network resources.

The identity validation restricts the devices from establishing multiple simultaneous connections with a given server. As our propose authentication scheme uses CoAP-based client and server interaction model hence, each client object is restricted to a single connection with a given server. This objective serves two purposes. First, CoAP is specifically designed for energy-constrained devices; hence energy should be utilized in an efficient manner. Second, the magnitude of any security breach has a direct relation with the number of established connections. However, connection restriction alone is not sufficient to achieve a robust secure solution. Each connection needs to be authenticated using some lightweight encryption scheme. Off course, ID validation and authentication cannot combat all types of threats in any communication network. However, they are still capable to tackle a subset of malicious threats.

Another challenging task in an IoT communication is the resource observation at the server. Each application has its own requirements in terms of data acquisition and resource observation. Some applications require periodic transmission while other requires continuous data flow. Yet, there are other applications which observe abrupt and sudden changes in the resource state. Thus, the IoT needs to be configured in a way to fulfill the requirement of each application.

The conditional resource observation is well suited to the energy-constrained devices of an IoT environment. Each client specifies certain conditions in the request message. Once the condition is fulfilled at the server end, it notifies the client about the current state of the resource. This reduces the number of transmitted packets. Thus, enhancing network lifetime by reducing the energy consumption along with the network congestion

Resource observation, secure authentication and communication need to be provided at a higher standard to ensure that the IoT devices and products are more robust and beneficial. Delivering the data intended for one device to a different and irrelevant device would harm the network resources in return.

III. Lightweight Mutual Authentication Scheme for Conditional Observation of Resources

In this section, we first present a brief overview of our authentication scheme followed by the actual data transmission

which is based on the conditional resource observation. Here, it is important to mention that our authentication algorithm and CoAP protocol are not two separate protocols; but we have added security features for the authentication in the CoAP to make it more robust, efficient and secure against various types of threats. Implementing two or more protocols in the IoT domain is not a feasible and optimal solution in terms of interoperability, scalability, resource utilization and hardware and software complexity. In short, its a two-step process: authentication and communication.

A. Lightweight CoAP-based Authentication Scheme

Like any communication network, preserving resources in the IoT domain is of utmost importance supporting greater operational cycles. Fabricating and spreading out malicious data in the network will risk the whole network traffic flow. Consequently, each device in the network will end up with malicious copies of the data. Both the client and the server have equal chances to be compromised in the network. Hence, it is important to authenticate the identities and integrity of the communicating parties.

As CoAP is a lightweight alternative of the HTTP protocol for the resource constrained devices, hence simple but robust authentication schemes need to be developed in order to prolong the network lifetime. Here, we have proposed an authentication algorithm in view of the above discussion. The algorithm is simple in computation but it can be a robust alternative to the Datagram Transport Layer Security (DTLS) due to its ease of implementation, infrastructure and complexity. Both the client and server challenge each other to authenticate themselves. The authentication mechanism completes using four handshake messages between any client and the server. Each message is comprised of 256 bits except the initial handshake authentication request of the client. Hence the total connection overhead incurred during the authentication session is less than 1024 bits, which is well suited to the IoT-based devices.

In this scheme, the authentication is completed using the 128-bit Advanced Encryption Standard (AES). We believe that the 128-bit encryption is sufficient for the energy-constrained devices participating in the IoT paradigm. Both AES-192 and AES-256 bit encryption schemes are highly computational and time-consuming tasks requiring sophisticated hardware and software platforms, which is not the case with majority of the energy-constrained objects.

Each client shares a pre-shared key, Y_i of 128-bits with the server before authentication initiation commences. Each device has a unique ID associated with it, which enables the server to perform a table look-up for the identity validation. The server maintains a pool of pre-shared keys which are associated with the unique ID of each client. Any pre-shared key, Y_i is known only to the server and the client with whom the communication is to be established. The session initiation is validated based on the match in the look-up table. If the client ID is not validated, the session initiation fails. Figure 2 shows the Key-ID pairs in the table maintained by the server.

Device ID(i)	1	2	3	4	5	6	.....	n
Pre-shared Key Y_i	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	.....	Y_n

Fig. 2. Key-ID Pairs in the Server Table

Session initiation is only the first step in the proposed authentication scheme. The ID matching in the table only allows both the parties to communicate with each other for exchanging the session key. However, the actual authentication process is completed using four handshake messages as shown in Figure 3. The handshake mechanism is explained below in four simple steps.

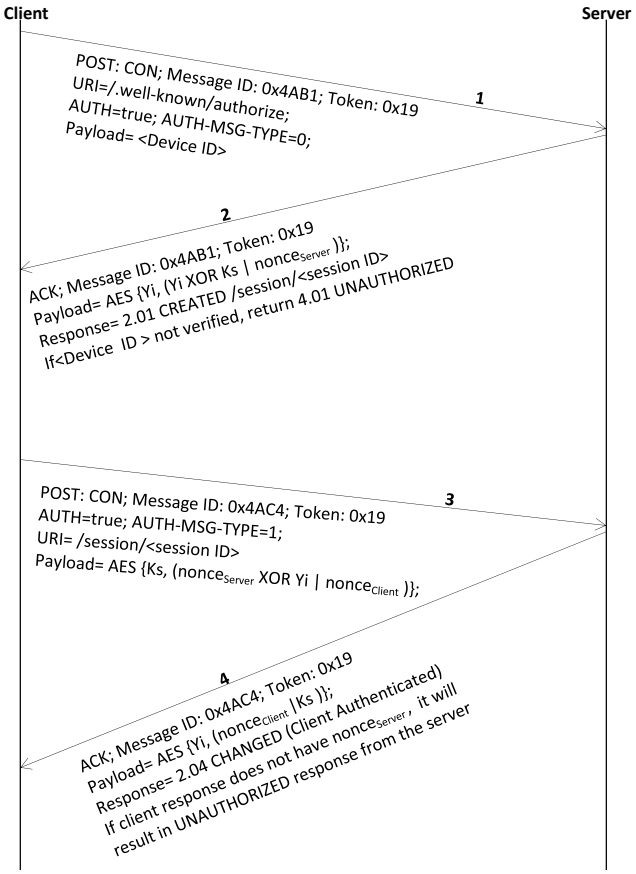

Fig. 3. Four-way Authentication Handshake

1) Session Initiation: Each client sends a request message to the server. This message is of the type CON having POST method applied to it, which is used to create or update a resource. Each message has a specific token associated with it. The token is used for correlating the request with a matching response. Also, each message has its own ID for unique identification. Every client maintains and monitors a queue of all transmitted CON messages. The CON message is retransmitted if an acknowledgment is not received in a prespecified duration or else, the message timeouts. The message carries two options, the Auth and the Auth-Msg-Type. The values of these options indicate the type of operation performed on a resource at the server. The client request is directed towards the resource identified by a URI as shown in Figure 3. In this figure, /authorize is the resource at the server. The parameters Auth=true, Auth-Msg-Type $= 0$ and $/$authorize informs the server that the client request is for the session initiation.

2) Server Challenge: The server retrieves the device ID from the message payload. This enables the server to find a matching pre-shared key associated with each client. Once a match is found, the server responds back with an encrypted payload using the AES algorithm. The server generates a pseudorandom number, nonce_{Server} and a potential session key, K_s. Each one is of 128-bits and the nonce_{Server} is used only once in the entire authentication process. At this stage, an encrypted payload is generated. First, XOR operation is performed on Y_i and K_s. Next, the result is appended with the nonce_{Server}, which is finally encrypted with Y_i as shown in the equation 1.

$$E_{SP} = AES\{Y_i, (Y_i \veebar K_s | nonce_{Server})\} \tag{1}$$

In this equation, E_{SP} is the key used for encrypting the payload generated by the server and Y_i is the key with whom

the payload can be encrypted. Here, Y_i is the key used for encrypting the payload. As it is common to both the client and the server. The XOR operation (represented by $\veebar$) is the most commonly used operation in the symmetric cryptography for encryption and decryption as it yields the original information if it is used twice to encrypt and decrypt the same message.

Next, the encrypted payload is transmitted to the client as a challenge which needs to be decrypted in order to establish an authentication session. The session initiation results in the creation of a resource at the server. In this case, the resource is the session created by the server as depicted by a temporary session ID in the response URI. If the server is unable to verify the ID of the client, the session is terminated and the handshake mechanism does not proceed any further. This is the case when the client is an attacker who wants to initiate a session with the server.

3) Client Response and Challenge: Once a client receive the challenge, it needs to decipher the encrypted payload in order to retrieve the potential session key, K_s. If the client is able to successfully decrypt the payload, it will have the correct $nonce_{Server}$ and K_s. Each client uses it's pre-shared key, Y_i to decrypt the payload. Upon successful deciphering, the client has authenticated itself and now the server also needs to authenticate itself. The client generates a new encrypted payload. First, an XOR operation is performed on $nonce_{Server}$ and Y_i. Next, the resultant is appended with $nonce_{client}$ and encrypted with K_s as shown in equation 2.

$$E_{CP} = AES\{K_s, (nonce_{Server} \veebar Y_i | nonce_{Client})\} \quad (2)$$

Here, E_{CP} is the encrypted payload generated by the client, $nonce_{Client}$ is the pseudorandom number generated by the client. Like $nonce_{Server}$, the $nonce_{Client}$ is generated only once in any specific authentication session. The value of Auth is set to $true$ while Auth-Msg-Type is set to 1 in order to inform the server that the encrypted payload in the server challenge was successfully deciphered. At this stage, K_s has securely been transmitted to the client.

4) Server Response: Finally, the server checks the encrypted payload in the client challenge. Upon observing the $nonce_{Server}$ in the client response, the server realizes that the client has successfully authenticated itself. Now, the server decrypts the payload and retrieves the $nonce_{Client}$ from it. Next, it creates a payload by embedding the $nonce_{Client}$ in it and appending K_s. The payload is then encrypted with Y_i as shown in equation 3.

$$E_{SP} = AES\{Y_i, (nonce_{Client} | K_s)\} \quad (3)$$

As the client has already authenticated itself, hence the status of the resource changes to $Authenticated$ at the server. Next, the encrypted payload is transmitted to the client. Upon reception, the client decrypts it and observe the $nonce_{Client}$ in it, which indicates that the server has also been able to decipher the challenge successfully. Hence, both the parties have authenticated themselves and are authorized to exchange the data packets. At this stage, both the client and server have agreed upon a common session key, K_s which was securely transmitted.

The format of the Auth and Auth-Msg-Type options is shown in Figure 4.

No.	C	U	N	Name	Format	Length	Default
TBD	X	X	-	Auth	empty	0	(none)
TBD	X	X	-	Auth-Msg-Type	uint	1	(none)

Fig. 4. Authentication Options Format

Both these options are critical and unsafe-to-forward. Critical or elective options are related to the endpoints while safe or unsafe-to-forward are used in the proxy context. If an endpoint is unable to understand a message having a critical option, it must return a $Bad\ Option$ response to the sender. It is mandatory to understand the critical options in a message before forwarding it to other devices. Unsafe-to-forward is having the same meaning as critical option and is used only in the context of the proxies. Both these options are yet to be assigned numbers by the Internet Assigned Numbers Authority (IANA) [3].

Once the connection request is authorized, the actual data transmission commences. Each authenticated client registers itself with the server for the resource observation.

B. Conditional Resource Observation in Internet of Things

We have implemented our authentication scheme for the conditional resource observation using the CoAP protocol. Our implementation consists of four clients which conditionally observe the temperature readings at the server. Each client is associated with a specific application which has its own requirements for the temperature notifications. A server Netduino Plus 2 board continuously monitors the temperature readings. An air conditioner, a room heater, a tap valve and window blinds are the four applications in our proposed scheme. For visibility purpose, we have shown only two applications in Figure 5.

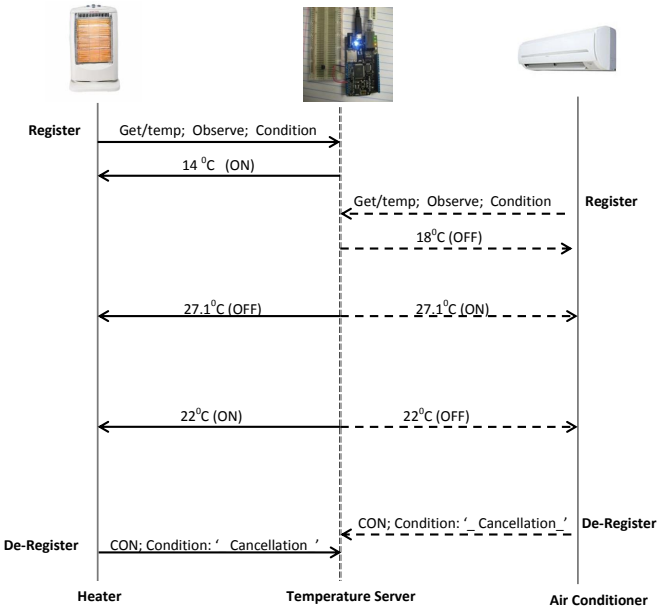

Fig. 5. Conditional Resource Observation

Each client registers itself with the server by sending a request message. This message contains a conditional option along with the observed option. Here, a GET method is used in the request to retrieve the representation of a resource. The server verifies the presence of these options in the request message. If present, the client is registered as an observer in the internal list maintained by the server. We have modified the conditional option in our implementation based on each application's requirement. An air conditioner client requests the server for notification whenever the state of the resource is greater than 25^0C. When the temperature readings are lower than this value, the electric heater will be notified. The tap valve and the wall blinds have different requirements for the notifications as compared to the heater and the air conditioner. The server notifies the tap valve whenever the temperature readings are greater than 35^0C. Upon reception of the temperature readings, the status of the tap valve changes

[3]https://www.iana.org/

from 0 (OFF) to 1 (ON) in order to water the plants in a pot. When the temperature readings are greater than 30^0C, the wall blinds are notified to turn off.

Our scheme connects four different types of physical objects in an IoT communication system. Interestingly, all these objects perform different operations on the same resource. As their application requirements differ from each other, hence their mode of observation also differs. As miniature sensor nodes are involved at the core of the communication, hence their resource-constrained nature needs to be preserved. The conditional resource observation enhances the battery power of these nodes by decreasing the number of transmitted notifications. In our scheme, the server notifies the clients whenever certain conditions are satisfied. However, it does not transmit any other packets as long as the condition is true. For example in Figure 5, initially, the server notifies the room heater with a temperature reading of 14^0C. As this value is less than 25^0C, hence the sensor responsible for the control of room heater turns it ON. Next, if the temperature readings changes and they are less than 25^0C, the room heater will not be notified. After all, any reading less than 25^0C would mean to turn on the heater, which is not required as the heater is already in the ON state.

If a client wants to deregister from the conditional resource observation, it transmits a CON message to the server. In this message, the conditional option type is set to $\_Cancellation\_$. When the server observes this value, it deregisters the client and removes it from the observer list.

For conditional resource observation, both conditional and the observe options are mandatory in a registration request message. The observe option registers a client for resource observation, while the conditional option informs the server about the notification criteria. In order to understand the internal mechanism for the notification, it is important to have an understanding of the conditional option format. It is 1 to 5 bytes in length [5] consisting of a 1 byte mandatory header and 0 to 4 bytes value. It is elective and unsafe-to-forward as shown in Figure 6.

Number	Elective	Unsafe	Format	Length	Name
18	Yes	Yes	uint	1-5 Byte	Conditional

Fig. 6. Conditional Observe Option

In conditional observation, the header plays a vital role in specifying the criteria for the resource observation. The header consists of three fields, a 5-bits type, 1-bit reliability flag and a 2-bit value. The type field results in 32 various combinations for condition specifications in a request message. Till now, only 10 condition types are known. For example, cancellation, minimum response time, maximum response time, value $<>$ and periodic are some to mention in this context. Their IDs are set in the type field of the header. Its format is shown in Figure 7.

7	6	5	4	3	2	1	0
Type					Flag	Value	

Fig. 7. Header Format

Each client needs to register itself only once with a given server for a resource observation. An attacker might register itself multiple times for a resource and will be able to conduct various types of DOS attacks. However, our authentication scheme prevents the attackers from registering multiple times with a given server. This is due to the fact that each server, first validates and authenticates the identity of a device and later on allows it for the resource registration.

IV. Experimental Evaluation

In this section, we have provided the preliminary evaluation results for our scheme. First, any client and server participating in an IoT communication are authenticated by validating their IDs. We have used CoAPSharp[4], an open source library for authentication and conditional resource observation. This library has very basic implementation of the CoAP protocol and only provides normal resource observation. Hence, we implemented our authentication scheme and application specific conditional option in it.

For the implementation, we first performed our evaluations on the emulators, which were then verified and implemented on the NetDuino Plus 2 boards[5]. A temperature sensor, Dellas DS1820[6] was embedded on the NetDuino Plus 2 Board. The NetDuino board in the role of a server provides conditional specific resources to four different clients in our proposed approach. Each NetDuino Plus 2 board control an application, as discussed in the previous section. Hence, our setup consists of a total of five boards, a server and four client NetDuino boards.

The server and the clients authenticate each other before establishing a conditional resource observation relationship. Figure 8 shows a successful authentication of this interaction. In this figure, the server key is the potential session key which needs to be securely and successfully transmitted to each client. Upon successful decryption, the authentication process is completed. Here, the client key is the pre-shared secret key associated with each client.

```
The thread '<No Name>' (0x2) has exited with code 0 (0x0).
[SERVER] Started.
[SERVER] Key: 4F9DB1949D924031-8C77BE06276ECB25 Nonce1: 2F2515012EDB8CE0-5A02705303A1544C
Nonce2:
[CLIENT] Key: 16BBE8D16B4C00F8-3143F1D60DA5E97D Nonce1: 2F2515012EDB8CE0-5A02705303A1544C
Nonce2: 5E10A9012748BDDA-3FFDCFF6128F4056
[CLIENT] Replying to server challenge...
[SERVER] Access granted to client 16BBE8D16B4C00F8-3143F1D60DA5E97D
[SERVER] Key: 4F9DB1949D924031-8C77BE06276ECB25 Nonce1: 2F2515012EDB8CE0-5A02705303A1544C
Nonce2: 5E10A9012748BDDA-3FFDCFF6128F4056
```

Fig. 8. Successful Authentication Response

Figure 9 shows an unsuccessful authentication response. Here, the client is unable to decrypt the session key. Hence, the client is prevented from registering with the server for the resource observation. Failure to decrypt the session key prevents various types of attacks in an IoT environment.

```
'Microsoft.SPOT.Emulator.Sample.SampleEmulator.exe' (Managed): Loaded
'C:\Users\mian\Desktop\sources-20140528\sources\CoAPTest-Server\bin\Debug\le\CoAPTest-
Server.exe', Symbols loaded.
The thread '<No Name>' (0x2) has exited with code 0 (0x0).
[CLIENT] Started.
The thread '<No Name>' (0x2) has exited with code 0 (0x0).
[SERVER] Started.
[SERVER] Key: 4F9DB1949D924031-8C77BE06276ECB25 Nonce1: 4BAFCDE9430E5773-24E5095A6614BF17
Nonce2:
[CLIENT] Key: 6619DB083FA7049A-70F225566ED3847A Nonce1: 4BAFCDE9430E5773-24E5095A6614BF17
Nonce2: 6DFB243F1F64BE50-142C6CFE3382DAAF
[CLIENT] Replying to server challenge...
[CLIENT] Resource access denied.
```

Fig. 9. Unsuccessful Authentication Response

In Figure 10, the status of various physical devices registered with the server for conditional resource observation is shown.

Here, each device rely on the temperature readings of the server. We have specified various conditions for the notification messages to the server. Each device remains in a particular state (ON/OFF) and switches its state once a particular condition is satisfied. Various conditions specified for our experimental results are already explained in the previous section. Here, 0 represents OFF and 1 represents ON state.

In the above figure, we have provided the preliminary results. Currently, we are conducting extensive mathematical

[4]http://www.coapsharp.com/

[5]http://netduino.com/

[6]http://www.micropik.com/PDF/ds1820.pdf

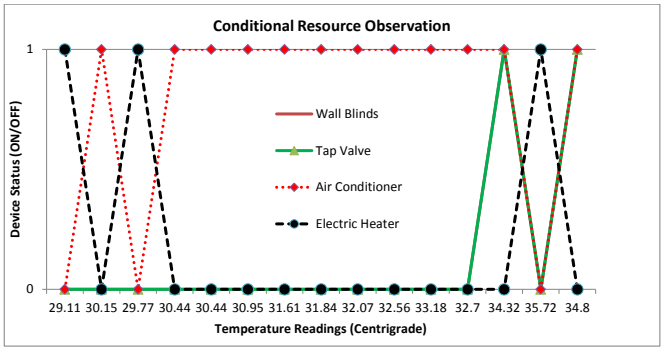

Fig. 10. Conditional Resource Observation

and experimental evaluation of our proposed scheme against the DTLS and PKI in terms of various performance metrics like the data rate, throughput, packet loss, latency and average battery power consumption.

V. Related Works

In this section, the related works from literature on authentication and resource observation is provided. As our work is specifically related to the IoT applications, hence an overview of the underlying application layer protocol is provided first. The protocol provides the platform for resource retrieval and manipulation in an energy-efficient manner.

The internet of today is based on the REpresentational State Transfer (REST) architecture using HTTP protocol [6]. The HTTP is a resource-oriented protocol requiring extensive memory and computational capabilities. However, the devices in an IoT environment are resource-constrained which render their capabilities to support the HTTP protocol. To provide RESTful services in the constrained networks, Internet Engineering Task Force (IETF) has formed a special working group known as Constrained RESTful Environments (CoRE) which is responsible for developing the standards and lightweight protocols [7]. The Constrained Application Protocol (CoAP) is one such product of this working group.

The CoAP is an application layer protocol which inherits a subset of HTTP features to suit the requirements of constrained networks [8]. It uses a request/response interaction model to exchange the resources between any client and server. Each client sends a request to the server to provide the representation of resources. One or more resources may reside on the server. All interactions are managed using the Uniform Resource Identifiers (URIs) to fetch resources from the server, which are provided in various formats to the clients. The protocol supports four types of messages: Confirmable (CON), Non-Confirmable (NON), Acknowledgement (ACK) and Reset (RST). As CoAP is based on the UDP protocol, hence messages may arrive out of order, missing or corrupted. To provide reliability, CON message type is used in the request which needs to be acknowledged by the recipient.

Each client sends a registration request message to the server for resource observation using CoAP protocol [9]. The request contains a message ID, an observe option and a GET method for the resource retrieval. The observe option informs the server to register the client for any future notifications. The request message is of the type CON which needs to be acknowledged by the server. The server registers the client and notifies it whenever the state of the resource changes. In each notification message, a 24-bit sequence number for the observe option is appended which allows the client to determine the freshness of the resource. Also, this sequence number enables the client to reorder the notification messages. To determine freshness of an incoming notification, each client implements a simple logic as shown in equation 4.

$$\begin{aligned} &(V_1 < V_2 \quad \text{and} \quad V_2 - V_1 < 2^{23}) \;\; Or \\ &(V_1 > V_2 \quad \text{and} \quad V_1 - V_2 > 2^{23}) \;\; Or \\ &(T_2 > T_1 \;\; 128 \;\; seconds) \end{aligned} \tag{4}$$

Here, $V1$ is the sequence number of the observe option in a previously received response notification and, $V2$ is the sequence number of the latest notification. Assume, $T1$ is the time when $V1$ was received and, $T2$ is the time when $V2$ is received. The client determines the freshness of a newly received notification by using the logic of equation 4. The above condition specifies that either $V2$ is a larger sequence number than $V1$ or $V2$ is smaller than $V1$ due to the rollover of arithmetic numbers [10]. Alternatively, if a longer time has elapsed since $V1$ and $V2$ were received, we simply ignore them and accept a new value of 128 seconds. This new value was arbitrarily chosen to be a number sufficiently larger than the maximum latency [8].

The drawback of normal resource observation is that the server notifies the client whenever the state of the resource changes. It is a burden on the energy-starved devices in an IoT environment. Each application has its own requirements for notification messages. An electric heater might not be interested in temperature readings greater than 35^0C. However, such readings are critical for the operation of an air conditioner. Hence, each device needs to specify its own conditions in the request message for the transmitted notifications. Each client appends a conditional option along with the observe option in the registration request message [5]. This combination enables the server to uniquely distinguish the registration message as a conditional observation request.

In an IoT environment, the resources need to be observed in a secured fashion among the trusted and authenticated devices. In [11], the authors highlighted various security challenges faced by an IP-based IoT network. Secure and lightweight communication protocols need to be designed to suit the resource-constrained nature of the physical objects. Security provisioning in small devices is a challenging task because resource limitation restricts to secure all layers on individual basis. If the application layer is secured while the data link and network layer are left unsecured, it will expose these layers to various types of attacks, common to the internet of today. Securing the data link and network layer at the expense of application layer will result in inter-application vulnerable attacks.

In [12], the author has proposed RSA algorithm. The RSA uses Public Key Infrastructure (PKI), where a pair of public and private keys is used. PKI is a resource consuming cryptography scheme, hence it does not suit well to the energy-constrained nature of certain devices in an IoT environment. In [13], the authors have proposed Server-based Certificate Validation Protocol (SCVP). This protocol enables a client to delegate a certificate validation to the trusted server. The proposed approach increases the communication overhead and does not suit well on a per-handshake basis in an IoT communication network.

In the internet, certificate validation and PKI are the robust and frequently used authentication schemes. However, they are computationally complex, resource consuming and requires proper configuration to be implemented in an IoT domain. The implementation of key-pairs restricts many small devices from using these schemes. As CoAP is based on the UDP protocol, hence Datagram Transport Layer Security (DTLS) is the obvious choice for authentication [14]. However, the DTLS implementation with full PKI is not a resource-optimal solution for the constrained devices. In [15], the authors have

proposed an authentication scheme to establish a unicast communication channel. The proposed approach uses the symmetric encryption scheme which reduces energy consumption and computation. The authors claim that DTLS can be configured to develop an energy efficient authentication scheme. However, they have not validated their claim.

In view of the above discussion, we have proposed a lightweight scheme using a single key for the authentication purposes. It incurs a small connection overhead and is computationally simple and robust. The handshake mechanism is used for client and server authentication. Once authenticated, the clients register themselves for observing resources at the server. Our scheme restricts the malicious nodes from establishing multiple connections with the server at a given time. Each client is allowed to establish only a single connection with the server in order to fairly utilize its limited resources. Authentication ensures that a legitimate server transmit resource representation to the legitimate clients. After all, we do not want a compromised server to transmit false readings involving critical applications in a sensor based network.

VI. CONCLUSION

The Internet of Things (IoT) is a novel concept which integrates physical objects in the internet. There are lots of speculations about the future of today's internet. The IoT is gaining popularity among researchers in academia and industry as well. Various manufacturers and vendors have made optimistic future forecasts about the growth of the IoT products. Various lightweight protocols have been developed in order to establish communication among various physical objects. However, in the absence of a unified standard, it's difficult to provide seamless and an interoperable communication among these devices. Hence, compatibility is one such issue that needs to be addressed in order to transform speculations to reality.

Despite all these speculations, fewer efforts have been made to secure these products. More and more IoT products are reaching the market but unfortunately most of them lack secure features. As devices are having different attributes and specifications, hence secure solutions for the internet are not feasible for them. Although, sensors are involved at the core of the communication system in these objects, however, secure solutions for sensor networks are incompatible for these physical objects due to their underlying hardware and software platforms. In the literature, there exists little work on securing these objects, which is mostly in the IETF RFC drafts.

In this paper, we have proposed a lightweight authentication scheme which verifies the identities of the participating clients and servers in a CoAP-based IoT environment. A session key is exchanged between the communicating endpoints. Each client maintains a session key with a given server which guarantees that both the parties have been authenticated. Once authentication is completed, the clients register themselves with the server for resource observation. Each client specifies certain conditions for the notification messages based on their application requirements. The server notifies the clients once those conditions are fulfilled. This reduces the number of undesirable transmissions which enhances network lifetime and leverages congestion.

The proposed scheme is an effective solution against eavesdropping, key fabrication, resource exhaustion and denial of service attacks. However, it is not efficient against Sybil attack [16]. But again, no secure solution in the world can combat all types of attacks. In Sybil attack, a single malicious node poses multiple identities to the communicating devices at a given time. These identities are either fabricated or stolen by disabling the legitimate nodes of the network. Hence, a single physical device can harm multiple network resources. In Figure 1, the intruder will be a Sybil device, if it poses three different identities to the smart phone, internet and the sensor network. Apart from Sybil, various other threats like wormhole, sinkhole and selective forwarding are yet to be explored in an IoT domain.

Despite all the speculations and future forecasts, provisioning of the security features will always remain a major concern for the IoT products and devices. We should always keep in mind the consequences of security breaches like what will happen in the event, when our physical devices are hacked by someone with the ability to turn off our water supply, take the control of our cars, unlock the doors of our homes from thousands of miles away or generate the false fire alarms. These situations encourage security researchers to explore further and develop solutions to combat security threats of the future involving IoT.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] M. Kovatsch, "Firm firmware and apps for the internet of things," in *Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications*. ACM, 2011, pp. 61–62.

[3] D. R. Raymond and S. F. Midkiff, "Denial-of-service in wireless sensor networks: Attacks and defenses," *Pervasive Computing, IEEE*, vol. 7, no. 1, pp. 74–81, 2008.

[4] K. Pelechrinis, M. Iliofotou, and S. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *Communications Surveys Tutorials, IEEE*, vol. 13, no. 2, pp. 245–257, Second 2011.

[5] S. Li, J. Hoebeke, and A. Jara, "Conditional observe in coap," *Constrained Resources (CoRE) Working Group, Internet Engineering Task Force (IETF), work in progress, draft-li-core-conditional-observe-03*, 2012.

[6] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 2, pp. 115–150, 2002.

[7] A. McGregor and C. Bormann, "Constrained restful environments (core)," Tech. rep., Internet Engineering Task Force (IETF), Tech. Rep., 2012.

[8] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained application protocol (coap), draft-ietf-core-coap-13," *Orlando: The Internet Engineering Task Force–IETF, Dec*, 2012.

[9] K. Hartke, "Observing resources in coap," 2013.

[10] R. Elz, "Serial number arithmetic," 1996.

[11] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, 2011.

[12] E. Milanov, "The rsa algorithm," *June 2009*, 2009.

[13] T. Freeman, A. Malpani, D. Cooper, and R. Housley, "Server-based certificate validation protocol (scvp)," 2007.

[14] K. Hartke and O. Bergmann, "Datagram transport layer security in constrained environments," 2012.

[15] A. Bhattacharyya, A. Ukil, T. Bose, and A. Pal, "Lightweight mutual authentication for coap (wip)," 2014.

[16] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 259–268.