CoAP-Based Request-Response Interaction Model for the Internet of Things

Fazlullah Khan[1(✉)], Izaz ur Rahman[1], Mukhtaj Khan[1], Nadeem Iqbal[1], and Muhammad Alam[2]

[1] Abdul Wali Khan University Mardan, Mardan, Pakistan
{fazlullah,izaz,mukhtaj.khan,nikhan}@awkum.edu.pk
[2] Instituto de Telecomunicações, Aveiro, Portugal
alam@av.it.pt

Abstract. The Internet of Things (IoT) is a broad vision that incorporate real-wold devices from everyday life. These objects coordinate with each other to share the information gathered from phenomena of interest. IoT is a broad term and has attain popularity with the integration of Cloud Computing and Big Data. The partnership among these technologies is revolutionizing the world in which we live and interact with different devices. On the down side, there are lot of speculations and forecasts about the scale of IoT products expected to be available in the market. Most of the products are vendor-specific and as such are not interoperable. They lack a unified standard and are not compatible with each other. Another major issue with these products is the lack of secured features. Albeit, IoT devices are resource-rich, however, they are not capable to communicate in absence of embedded sensor nodes. The presence of resource-constrained sensors in the core of each IoT device make it resource-starving and as such require extremely lightweight but secured algorithms to combat various attacks and malevolent entities from spreading their malicious data. In this paper we aim to propose an extremely lightweight mutual handshaking algorithm for authentication. The proposed scheme verifies the identity of each participating device because establishing communication. Our scheme is based on client-server interaction model using Constrained Application Protocol (CoAP). A 4-byte header, extremely lightweight parsing complexity and JSON based payload encryption make it a lightweight scheme for IoT objects. The proposed scheme can be used as an alternative to DTLS schemes, the one common nowadays for IoT objects.

Keywords: Internet of Things · Constrained application protocol · Mutual authentication · Resource-observation

1 Introduction

Technological advances in Micro-Electro-Mechanical- Systems (MEMS) and wireless communication has formed a solid foundation for sensor-embedded

J. Ferreira and M. Alam (Eds.): Future 5V 2016, LNICST 185, pp. 146–156, 2017.
DOI: 10.1007/978-3-319-51207-5_14

Internet of Things (IoT) [2]. The basic aim of IoT is to incorporate real-world physical objects by using unique addressing schemes [1]. CISCO has estimated that the number of such objects interconnected with each other and with Internet will surpass 50 billion by 2020[1]. These objects would be enabled to capture, compute and control the events, also known as phenomena of interest, occurring in the real-world [3]. Eventually, this fascinating concept of IoT will lead us to IoE, i.e., Internet of Everything, in which data, systems, objects and interacting processes will be part of it.

Integration of physical objects with the Internet is a challenging task. Security provisioning, an issue faced by the physical devices is of particular concern. This is because each physical object connecting with the Internet has distinguishing features and requirements. Without identity verification, a malevolent entity can easily gain access to a network and perform various malicious and harmful activities. Such malicious activities may include conveying falsified health readings to the doctors residing in distant locations, activation of fake fire alarms in an organization are few of the example in this context. Although, these security threats are highly vulnerable in nature and behavioral, however, very little have been done to secure the inter-connecting physical objects and their end-products. Because of that, the end-products of IoT available in the market are prone to a wide range of security breaches. As a result, Internet of Things (IoT) will eventually leads us to IoV, i.e., Internet of Vulnerabilities.

This paper aims to address the above issues by designing an extremely light-weight but highly secured and robust authentication scheme. The goal of our proposed scheme is to verify the identities of communicating objects in the IoT paradigm. Our proposed scheme works on the application layer of any physical object and uses a well-known Internet of Things (IoT) protocol, known as Constrained Application Protocol (CoAP) [7] for the network operation. This paper has two major objectives. It first authenticates the identities of the physical objects inter-connecting with each other. Each physical object, in a role of a client, communicate with a given server for authentication. Authentication is mutual because both the client and server mutually authenticate the identities of each other. Unless both entities are authenticated, a connection, i.e., a session will not establish between them. Once authentication is successful, the clients are eligible to observe the resources at a given server. Each server resides a set of resources which can only be observe by a legitimate client, i.e., the one which has been authenticated successfully. Each client has the ability to specify certain conditions to the server for resource observation. Such conditional specification not only enable a client but also the server to conserve their limited resources. Resources are only observe once the condition for observing a resource are fulfilled at the server end. These two objectives are vital for any robust and secured communication system. Conditional resource observation is highly essential in these networks because each object has its own requirement for data observation, data rate, memory availability and sleeping schedule. The latter attribute is because of the embedded sensor node at the core of each physical object. In

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

our proposed scheme, data flow between a client and a server only commences once a successful session is authorized.

The rest of this paper is organized as follows. In Sect. 2, related work is presented followed by the proposed scheme in Sect. 3. Experimental work and analysis are provided in Sect. 4. The paper in concluded and future research directions and gaps are discussed in Sect. 5.

2 Related Work

In this section, we present related work on mutual authentication and resource observation in the Internet of Things (IoT) paradigm. Today, the Internet of present is mainly based on REpresentational State Transfer (REST) architecture. The said architecture uses HTTP protocol [4] for its operation. HTTP, on the other hand, is a resource-consuming protocol which require ample amount of storage and computational resources. The real-world physical objects in an IoT paradigm are highly resource-starving due to the underlying embedded sensor nodes and as such lack the support for HTTP protocol. For the provisioning of RESTful services in any resource-constrained network, Internet Engineering Task Force (IETF) has come with an extremely lightweight protocol, known as Constrained Application Protocol (CoAP). This protocol is a lightweight version of HTTP, however, it is not an alternative of the latter. Our previous work on secure communication and architecture for wireless sensor networks can be studied in [5,6].

CoAP was designed in view of limited resources of the objects. This protocol allows the exchange of messages between resource-starving physical objects over resource-limited communication networks [7]. In the communication context, resource-starving objects are miniature devices which lack the support for processing speed, power, storage, available bandwidth and data rate. Such devices are often built using an 8-bit or 16-bit micro-controllers. In some case, the micro-controllers have an upper bound of 32-bit. Unlike conventional networks, the resource-limited networks lack the support for a fully functional TCP/IP stack. IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) is a well-known example of such networks. Instead of TCP, CoAP uses UDP at transport layer for flow control and session initiation and work alike HTTP to match requests with corresponding responses. Similar to web, IP addresses and port numbers are used to locate a resource residing on a given serve. Various RESTful URIs are used to provide access to the resources. Methods such as GET, POST, DELETE and PUT are used in similar fashion to HTTP. CoAP is not a replacement of HTTP protocol, however, it uses a small subset of commands and context of HTTP to optimize for Machine-to-Machine (M2M) exchanges. CoAP can be considered as a method for accessing and invoking various RESTful services exposed by physical objects, also known as Things, over a physical network. CoAP supports four different types of messages and their specification are defined in the CoAP-draft [7].

1. **CON**: CON represents a confirmable message which requires a valid response. The said response can either be positive or a negative acknowledgement. If in case, an acknowledgment is not received by the sender, the request is re-transmitted until all such attempts for transmission are exhausted. The re-transmissions attempts increase in a non-linear, exponential fashion.
2. **NON**: NON represents a non-confirmable message and is used for unreliable transmission such as a request for sensor readings which are observed periodically. In such transmission, if one reading value is missed, there is little impact on the overall reading. NON messages are not acknowledged and the response is mostly NON as well.
3. **ACK**: ACK represents a valid acknowledgement and is either piggybacked in the response or send as a separate message. ACK is sent in response to a CON message and contains information about an observed data. If ACK is lost, the response need to be send again with the same ACK by the server.
4. **RST**: RST generally represents a negative acknowledgement and is used when the server wakes up from sleep mode and lose the context of the previous state.

In the Internet of Things paradigm, the resources residing on a given server need to be observed in a secured manner. A wide range of security challenges faced by IP-enabled real-world physical objects are highlighted in [8]. In view of these challenges, extremely lightweight, robust and secured protocols need to be designed to meet the requirement of resource-starving sensor-embedded objects communicating over resource-constrained networks. Despite the presence of sensor nodes at the core of each object, the wide-range of security protocols available in literature for Wireless Sensor Networks (WSNs) are not applicable to these objects [9]. This is due to the fact that sensor-embedded physical objects have their own unique attributes and characteristics and as such does not suit the available protocols for WSNs. Any designed protocol for IoT need to be lightweight as well in view of the underlying resource-constrained sensor nodes in each object.

An RSA-based encryption algorithm for the IoT objects was proposed in [10]. The proposed scheme used a pair-wise key, i.e., a public key and a private key. The proposed scheme is, however, highly resource-consuming and require heavyweight and resource-intensive cryptographic suites. As a result, it does not meet the demands of resource-starving objects. A server-based certificate validation protocol was proposed in [11]. The said protocol enables one or more clients to delegate certificate validation to an entrusted server. However, the proposed protocol increases communication overhead and as such does not fit to the requirement of resource-constrained objects of an IoT. Certificate validation and PKI are well-known cryptographic and authentication schemes in the Internet. However, for the IoT, these schemes are highly complex in terms of computation, storage and as such require proper configuration to suit the objects interacting in an IoT requirement. Implementation of key-pair approaches restricts miniature sensor-embedded objects from utilizing these schemes. Data-gram Transport Layer Security protocol (DTLS) is an obvious choice for IoT objects because CoAP protocol uses UDP as a default and resource-saving scheme [12].

However, DTLS with full PKI is not an optimal choice for IoT objects. A symmetric key encryption scheme was proposed by [13] for authentication. The proposed scheme uses a single pre-shared secret for establishing a communication session. Although, the proposed scheme reduces energy consumption and computational resources, however, it has not been validated via experimental results.

In light of the aforementioned discussion, we propose an extremely lightweight authentication approach which uses a single key for authentication. Our scheme incurs very small overhead and is sufficiently simple in terms of computation and resource utilization. A 4-way handshaking approach is adopted to authenticate the interested clients and servers. Upon successful authentication, each client registers itself with the server to observe a resource, temperature readings in our case. Malicious clients are prevented to observer resources and from establishing connections with a server. Each client is restricted only to a single connection for fair utilization of resources.

3 CoAP-Based Request-Response Interaction Algorithm

In this section, a brief overview of our CoAP-based 4-way handshake mechanism for authentication is presented. It is important to note that our scheme and CoAP are not two separate protocols. Instead, we use our own security patch embedded in CoAP for authentication purpose to tackle various attacks.

Similar to any other communication network, resource preservation is a challenging task and is of utmost importance. Data fabrication by malicious entities and its spread over a network will jeopardize the traffic flow of the whole network. As a result of data fabrication, each object in the network will end up with large number of copies of the malicious data. Not only the client, but the server is also vulnerable to be compromised by a malevolent entity. Therefore, it is mandatory to authenticate the integrity and identity of both the parties, i.e., the client and the server.

In light of the above discussion, we have proposed a lightweight algorithm which uses the underlying operational model of CoAP. The proposed scheme can be use as a lightweight alternative to DTLS because its simple to implement, flexible in terms of complexity and infrastructure. Each client and a server challenge for mutual authentication using the exchange of four simple handshake messages. Each message comprise of 256 bits. The only exception is the first message, i.e., initial session initiation request. The small size of messages incur small overhead during the authentication procedure and causes less burden to the IoT objects. We have used Advanced Encryption Standard 128 bits, i.e., AES-128 for authentication and encryption. The four phases of our authentication schemes are *Session Negotiation*, *Server Challenge*, *Client Challenge and Response*, and *Server Response*.

Before session negotiation phase, each client shares with a server a 128 bit preshared secret Y_i. This is a pre-requisite phase which takes place well before any authentication commences. Each object has a unique identity (ID) associated with it which enables a server to look-up for that ID in its table. Y_i is known

only to the client and the server as it is pre-distributed before any authentication commences. The first phase, i.e., the session initiation or negotiation, is validated once a match is found by the server in the table. If the ID of a client is not present, the server will not proceed to session initiation. Figure 1 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. 1. Pre-shared secrets and IDs in server table

ID matching with the table only allows the server and a client to communicate with each other to exchange a session key. The actual authentication is completed only using the four handshake messages as shown in Fig. 2. The handshake procedure is explained in details in the following four phases.

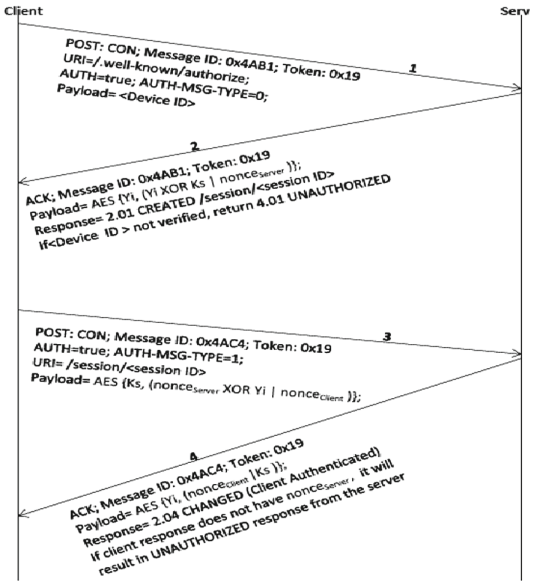

Fig. 2. Four-way authentication handshake

Session Negotiation. After a successful match of pre-shared secret, the actual authentication starts using the four-way handshake mechanism, i.e., four phases. In the first phase, a session is negotiated between the client and a server. Each client sends a request message to the server. This message is CON and the method is POST. The purpose of this message is to create a resource at the server. Each message contains a token which is used to correlate the CON request with a matching response (ACK). Also, each message has its own ID, to uniquely

identify it. Each client has the ability to maintain and monitor a buffer, which contains all transmitted request, i.e., CON messages to the server. If an ACK is not received within the specified duration, the CON message is re-transmitted. A CON message is also re-transmitted if the message timeouts. As shown in the Fig. 2, the session negotiation message also carries two options, i.e., Auth and the Auth-Msg-Type. URI is also present in the message which directs the client request towards a given resource. Here, in Fig. 2, /authorize is a resource residing at the server. In our scheme, the resource is temperature readings captured by a given server. The value of Auth = true, Auth-Msg-Type = 0 and /authorize is an indication to a server that the request is for session negotiation.

Server Challenge. Upon reception of session negotiation request at the server end, Object ID, is retrieved from the message payload. It enables a server to find a matching Y_i which is associated with a given client. If a match is found, then the server responds back with a payload which is encrypted using Advanced Encryption Algorithm (AES-128 bit). A pseudo-random number, a nonce ($nonce_{Server}$), and potential session key K_s are generated. All these parameters are of 128 bits. The nonce, on the other hand, is used only once by the server in the entire authentication mechanism. Using these parameters, the server generates an encrypted payload. An XOR operation is performed on K_s and Y_i. Then, the resultant is appended with $nonce_{Server}$ and is encrypted with Y_i. All these steps formulate Eq. 1.

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

In this equation, $E_{payload}$ is the resultant encrypted payload generated by the server as a challenge which need to be decrypted by a given client. Only a legitimate client can decrypt the payload by using the appropriate pre-shared secret.

Client Response and Challenge. When the client receive the encrypted payload, i.e., the result of server challenge, it needs to decrypt the said payload for the retrieval of K_s. If successful, the client will have the original K_s and $nonce_{Server}$. To decrypt the payload of server challenge, each client uses its unique Y_i, which is known only to a given client. Successful decryption of server challenge means that the given client has been able to authenticate itself. As our proposed scheme is based on mutual authentication, hence, the server also need to be authenticated. To do so, each client generates its own challenge, an encrypted payload, similar to the server challenge. For this, each client generates an encrypted payload of its own by using XOR operation as before. $Nonce_{Server}$ and Y_i are used as the two parameters for an XOR operation. The resultant of this operation is then appended to $nonce_{client}$ and encrypted with K_s. The detailed operation is shown in Eq. 2.

$$E_{payload} = AES\{K_s, (nonce_{Server} \ XOR \ Y_i | nonce_{Client})\} \tag{2}$$

Here, $E_{paylaod}$ is an encrypted payload generated by the client and $nonce_{Client}$ is a pseudo-random number, similar to $nonce_{Server}$, however, it is

generated by the client. Similar to nonce_{Server}, nonce_{Client} is generated and used only once in an authentication process. During this phase, Auth = *true* and Auth-Msg-Type = 1 literally means that the server should realize that this request is different than session negotiation and it means that the encrypted payload in a server challenge was successfully deciphered by the client. At this point, the potential session key, K_s has been securely transmitted to the given client.

Server Response. In this final phase, the server retrieves the encrypted payload from the client challenge. Upon observing nonce_{Server} in a client response, the server knows that the given client has been successful to authenticate itself. Now, the server also needs to decrypt the payload by retrieving nonce_{Client} from it. Upon successful decryption, the server creates a payload and embed the nonce_{Client} in it and appends K_s with it. This encrypted payload is encrypted with Y_i as shown in Eq. 3.

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

The client has already authenticated itself. So, the server changes the status of the temperature resource to *Authenticated*. Now, the encrypted payload is being transmitted to the given client. When the client receives it, it decryptes it and observe nonce_{Client} in it. By observing nonce_{Client} in the encrypted payload, the client realizes that the server has also authenticated itself. With this phase, both parties are mutually authenticated, are they are now ready to exchange data between themselves.

In our proposed scheme, we have created our own Options similar to the one CoAP uses. The formats of these two options, Auth and Auth-Msg-Type, are shown below in Fig. 3.

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. 3. Option formats

At this point of time, our proposed scheme only emphasis on authentication. We are still working on the actual exchange of data. For this we will specify various conditions for the exchange of data. Without successful authentication, any exchange of data is meaningless.

4 Experimental Evaluation

In this section, we have discussed the initial evaluation of our scheme. Before the initiation of communication, client-server authenticates each other by validating their IDs. For authentication and conditional resource observation, we

have applied an open source library, i.e. CoAPSharp. This library consist of basic CoAP protocol and offers normal resource observation. Therefore, we have modified the existing protocol with our authentication scheme and application specific conditional options.

In the implementation phase, we did our evaluations on the emulators first, and then confirmed and implemented on the NetDuino Plus 2 boards. A temperature sensor, Dellas DS1820 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 scheme. Each NetDuino Plus 2 board control an application, as discussed in the previous section. Hence, our test-bed is made-up of a total of five boards, a server and four client NetDuino boards.

Prior to setup a conditional resource observation relationship the client-server must authenticate each other. Figure 4 shows a successful authentication of this communication. 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. 4. Successful authentication response

Figure 5 shows an unsuccessful authentication response. Here, the client is unable to decrypt the session key. Therefore, the client is banned from registering with the server for the resource observation. Failure to decrypt the session key eliminates 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. 5. Unsuccessful authentication response

In Fig. 6, the status of various physical devices registered with the server for conditional resource observation is depicted.

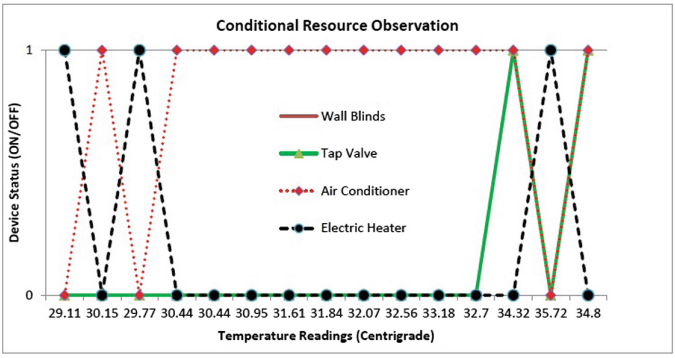

Fig. 6. Conditional resource observation

Here, each device relies on the temperature readings of the server. We have different and specific condition for the announcement of various messages to the server. Each device remains in a particular state (ON/OFF) and switches its state once a particular condition is fulfilled. Different 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 and experimental evaluation of our proposed scheme against the DTLS and PKI in terms of various performance metrics like the latency, packet loss, throughput, data rate, and average battery power consumption.

5 Conclusion

In an Internet of Things (IoT), not much efforts have been made for securing the IoT products available in the market. A large number of such products are reaching to the market, however, most of these products lack security features. Because, each object of an IoT has its own peculiar characters and has different attributes, the existing secured solutions available for the Internet are not feasible for apply to them. The presence of embedded sensors in each object does not mean that secured solution for WSNs are applicable to these networks because, of their own unique and distinguishing underlying hardware and software prototypes.

Our algorithm is highly efficient against key fabrication, resource exhaustion, eavesdropping and DoS attacks. However, it may not be efficient against Sybil attack [14]. But again, no secured solution in research can tackle all type of attacks. Despite the buzz and hype surrounding around Internet of Things, secured features will always remain a major concern for their products and objects due to their unique features and foremost we do not know what a real-world object will behave when it is connected with Internet. Such challenges

encourage academia and industry to explore in-depth and come up with various innovative solutions to tackle security loophole and vulnerabilities faced by these objects.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
2. Alam, M., Ferreira, J., Fonseca, J. (eds.): Intelligent Transportation Systems: Dependable Vehicular Communications for improved road safety, vol. 52. Springer, Heidelberg (2016). ISSN 2198-4128
3. Bormann, C., Castellani, A., Shelby, Z.: CoAP: an application protocol for billions of tiny internet nodes. IEEE Internet Comput. **16**(2), 62–67 (2012)
4. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Internet Technol. (TOIT) **2**(2), 115–150 (2002)
5. Khan, F.: Secure communication and routing architecture in wireless sensor networks. In: IEEE 3rd Global Conference on Consumer Electronics (GCCE), pp. 647–650. IEEE (2014)
6. Khan, F., Bashir, F., Nakagawa, K.: Dual head clustering scheme in wireless sensor networks. In: International Conference on Emerging Technologies (ICET), pp. 1–5. IEEE (2012)
7. Shelby, Z., Hartke, K., Bormann, C., Frank, B.: Constrained Application Protocol (CoAP), draft-ietf-core-coap-13. The Internet Engineering Task Force-IETF, Orlando (2012)
8. Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S.L., Kumar, S.S., Wehrle, K.: Security challenges in the IP-based internet of things. Wirel. Pers. Commun. **61**(3), 527–542 (2011)
9. Jan, M.A., Nanda, P., He, X., Liu, R.P.: PASCCC: priority-based application-specific congestion control clustering protocol. Comput. Netw. **74**, 92–102 (2014)
10. Milanov, E.: The RSA algorithm, June 2009
11. Freeman, T., Malpani, A., Cooper, D., Housley, R.: Server-based certificate validation protocol (SCVP) (2007)
12. Hartke, K., Bergmann, O.: Datagram transport layer security in constrained environments (2012)
13. Jan, M.A., Nanda, P., He, X., Tan, Z., Liu, R.P.: A robust authentication scheme for observing resources in the Internet of Things environment. In: IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 205–211. IEEE (2014)
14. Jan, M.A., Nanda, P., He, X., Liu, R.P.: A sybil attack detection scheme for a forest wildfire monitoring application. Future Gener. Comput. Syst. (2016)