Thread: An IoT Protocol

Ishaq Unwala
Computer Engineering
University of Houston Clear Lake
Houston, USA
Unwala@UHCL.edu

Zafar Taqvi
Computer Engineering
University of Houston Clear Lake
Taqvi@UHCL.edu

Jiang Lu
Computer Engineering
University of Houston Clear Lake
Houston, USA
LuJ@UHCL.edu

***Abstract*—*Thread* is a new IoT protocol released by Thread Group Inc. in 2015. More than 200 companies are currently members of the Thread Group Inc. and supporting this protocol. This paper provides an overview of the *Thread* protocol and discusses all the network layers. In the physical layer's overview we discuss the requirements for wireless communication. In the data link (MAC) layer we discuss the MAC link establishment (MLE) datagrams. We discuss use of 6LoWPAN addressing mode, which is a low power version of IPv6. In the network layer we discuss communication links and routing. In the transport layer we discuss the use of CoAP protocol and DTLS. In the security section we discuss setting up Commissioning agent and adding *Thread* devices to the Private Area Network.**

Keywords: Internet of things, IoT, Security, CoAP, IEEE 802.15.4, Thread, Threadgroup.

I. INTRODUCTION

Internet-of-Things (IoT), a term that was introduced by Kevin Ashton in 1999 [1], has grown considerably in recent years. IoT refers to Internet connected devices, these devices range from simple sensors and actuators to Cloud computing. IoT is expected to play an increasingly important role in our daily life, in future information, and soon many services will be delivered by IoT devices. IoT is being used to implement smart cities, smart workplaces, smart cars and smart homes. By end of 2017, 8.4 billion IoT devices are expected be installed and IoT related spending is expected to be $2 trillion [2]. The growth of IoT is projected to continue at a fast pace. Garner in press release [2] expects, by 2020, 20.4 billion installed IoT devices and spending of $2.9 trillion.

A. Internet of Things

An IoT system consists of sensors, actuators, communication and computing resources. A typical IoT is shown in Figure 1. An IoT system consists of a Private Area Network (PAN), an interface to the Internet through a gateway, and computing resources.

Each IoT protocol system specifies the requirements for the PAN, gateway(s) and Cloud computing. The PAN specification includes physical connectivity, topologies, security requirements, and gateway requirements.

Rest of the paper is organized as follows. Section II provides a short introduction to the *Thread* protocol. In Section II Subsection A we discuss the *Thread* Private Area Network (PAN), its requirements, features, and network topology. In Section II Subsection A through Subsection F we provide details of communication layers of *Thread*: Physical layer, Data link (MAC) layer, 6LoWPAN addressing, Network layer and finally the Transport layer. In Section II Subsection G we discuss the security and device commissioning aspects of the protocol. And finally, we conclude with some thoughts on the *Thread* protocol in Section III and some ongoing research work in Section IV.

II. THREAD PROTOCOL

Thread Group, Inc. was founded by seven companies, ARM (Softbank), Big Ass Fans, Freescale (NXP), Nest Labs (Google), Samsung, Silicon Labs, and Yale Locks. The *Thread* specification v1.0 was published in 2015 by Thread Group, Inc. [23]. Currently, Thread Group, Inc. has more than 200 members worldwide. Consumer level devices based on *Thread* are not yet readily available, although Nest thermostat and some *Zigbee* devices are *Thread* compatible. *Thread* is specifically designed for Home Automation and thus supports a wide variety of home use applications including, appliances, access control, climate control, energy management, lighting, safety, and security.

Thread is intended to be an open standard, and is built by incorporating many current standards and proposed standards (Request For Comments - RFCs), Figure 2. A complete specification is available from Thread Group, Inc. Although consumers currently cannot purchase *Thread* branded devices, *Thread* hardware with radios and software libraries are available for research and development. Currently, in market Nest thermostat and many *Zigbee* branded devices have a *Thread* compatible hardware.

This paper uses the *Thread* specification, v1.1.1 2017.

2166-5478/18/$31.00 ©2018 IEEE
DOI 10.1109/GreenTech.2018.00037

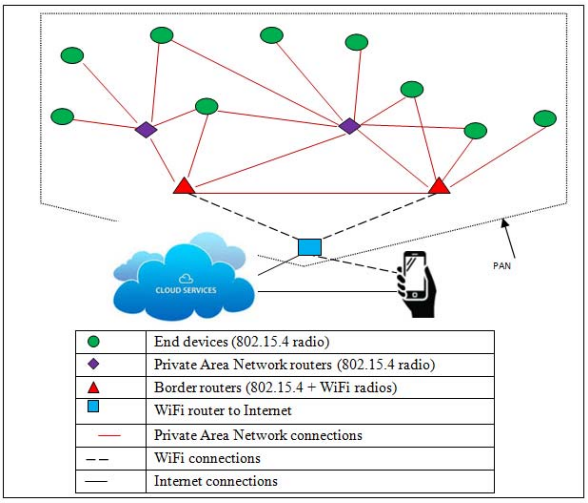

Figure 1. IoT system

A. *Private Area Network*

Private Area Network (PAN) is a user local area network consisting of sensors, actuators, computing, and communication hardware. The PAN is only connected to the Internet through a gateway. Each IoT protocol, *Z-Wave*, *ZigBee*, *Thread*, *IoTivity*, *AllJoyn*, and others, have their own specific PAN design requirements.

Before any device can be part of a PAN it must be commissioned. *Thread* uses MeshCoP (Mesh Commissioning Protocol). The MeshCoP protocol allows for securely authenticating, commissioning, and joining new-untrusted radio devices to a mesh network. More details on this are presented in Section II Subsection G.

A example of *Thread* PAN is shown in Figure 1. A PAN network provides the following features:

Node. *Thread* PAN consists of edges and nodes. Each node device has certain capabilities, some nodes are capable of connecting the PAN to the Internet, some nodes can only transmit to the nearest neighbor, and other nodes facilitate relaying data from one node to another. These node types are recognized by *Thread*, End Device (ED), Router Eligible End Device (REED), Full End Device (FED), Full Thread Device (FTD), Border Router device (BR), Leader router, Sleepy End Device (SED), Minimal End Device (MED), and Minimal Thread Device (MTD). Node types are transformable, depending on capabilities of the node device and the PAN requirements. For example FTD which is acting as an ED may request to become the router, or a router can become a REED.

Topology. Network topology links individual nodes together in an organized manner to allow routing. A number of topologies are used in IoT PAN including Star, Point-to-Point and Mesh. One of the requirements for *Thread* PAN was to have no single point of failure. *Thread* has selected Mesh network topology for its PAN. Mesh topology was chosen for its high resilience, scalability, robustness and ability to self-heal in case of a component failure.

Communication links. *Thread* PAN uses wireless radios as communication links of the network. The *Thread* PAN wireless radios operate on 2450 MHz frequency and are based on IEEE 802.15.4 standard. Further discussed in Section II Subsection B.

Routing. Each node can transfer data to its nearest neighbor, so data hops across multiple nodes arrive at their destinations. All routing links to neighboring nodes are discovered and automatically configured by *Thread* software. Further discussed in Section II subsection E.

Device identification. Each node in *Thread* PAN has an IPv6 address. The IPv6 address is compressed using 6LoWPAN. Further discussed in Section II Subsection D.

Send/receive node data. All data links are encrypted either at Data Link layer and/or the Transport layer.

Computing resources. All *Thread* PAN nodes require some computing resources to form data packets, encrypt/decrypt data, perform actions based on requests, setup transmission, etc.

Ubiquitous service. *Thread* PAN, like other IoT protocols, offers always avaliable service to its users.

Semantics. Software processing in the Cloud analyzes the sensor and input data, along with other information, and provides a context-aware response for the IoT system.

Gateways. *Thread* PAN can have multiple gateways, i.e. border routers, to the Internet. These multiple gateways add redundancy and remove a single point of failure. Border routers have both a PAN radio for connection to PAN nodes and either wired or wireless channels to connect to the Internet and Cloud servers.

All these features in the PAN are provided under many practical constraints.

Low Power. Lower power operation is a must for IoT devices, as many devices (e.g. SED type) are battery operated. Even if the IoT device is connected to the power grid, low power is an important consideration due to large number of these devices being used with round-the-clock operation.

Low Cost. If we want the IoT to be ubiquitous, it is important to have a low initial cost and a low maintenance cost devices.

Security. IoT security is serious issue. The security of the PAN and secure operation in the Cloud is critical. This is further discussed in Section II Subsection G.

Communication challenges. The PAN operates in a non-ideal indoor environment where there may be many obstacles in the communication path. These obstacles can change position, new ones can appear and old ones can disappear at anytime. Mobile IoT devices are allowed to drop-out or rejoin the network without notice. Due to this unreliable communication, *Thread* PAN uses DTLS, which assumes unreliable Transport layer. In addition IoT devices have a weak transmission signal due to low power requirements, making communication more challenging. IoT transmission packets are small, eliminating the opportunity to include redundancy for fault tolerance.

User interface. Most of IoT devices lack a user interface, e.g. an IoT light bulb, or have a very primitive user interface, e.g. an IoT light switch. To overcome this deficiency, IoT devices have the ability to connect to GUI-rich devices, such as a computer, a smart tablet, a smart phone, or a voice controlled device for commissioning and operation.

B. PHY (Physical) layer

All *Thread* devices support an interface conforming to PHY (Physical) specification defined in IEEE 802.15.4 (2006). Although IEEE 802.15.4 covers specification for 2450MHz and 868/915 MHz, *Thread* does not support 868/915 MHz. *Thread* only requires support for IEEE 802.15.4 specifications relating to 2450 MHz. *Thread* uses O-QPSK modulation and has a data rate of up to 250 Kbps. *Thread* radio signals have a maximum range of 30m per hop, with a default hop limit of 36 hops, and an ability to connect 250+ devices in a *Thread* PAN. IEEE 802.15.4 certified radios for *Thread* are ready available from vendors.

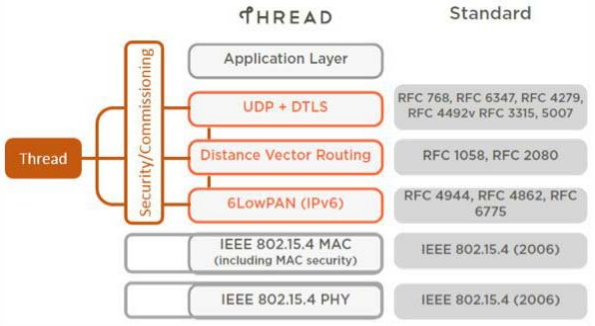

Figure 2. *Thread* protocol stack IoT system [23]

C. MAC (Data link) layer

Thread devices must also implement a subset of Media Access Control (MAC) specified in IEEE 802.15.4. All *Thread* devices with routing capability must be able to act as a MAC Full Function Device (FFD). All *Thread* devices without the routing capability must be able to act as a MAC Reduced Function Device (RFD).

Thread device must have the following MAC capabilities:

- Employ Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA).
- Have active and Energy Detect (ED) scan types.
- Be able to generate and receive following MAC frames
    - MAC data and Acknowledgement frames.
    - Data request and Beacon request frames
    - Beacon frame when requests received by FFD
- Provide a reliable link between two MAC entities
- Implement MAC frame security based on PAN setting

Thread MAC devices are NOT allowed to participate in the following activities:

- Operate in periodic Beacon-enabled mode
- Guaranteed Time Slots mechanisms
- Generating or interpreting following command frames:
    - Association Request
    - Association Response
    - Disassociation Request, PAD ID conflict
    - Orphan Notification
    - Coordinator Realignment
    - FDD PAN Coordinator mode
- Addressing mode 00 for Data or MAC command frames
- Any other MAC frame security options except as mentioned in PAN settings.

MAC Link Establishment (MLE) datagram can be secured or unsecured. Secured MLE is indicated with an initial byte value of 0 followed by an auxiliary header byte. For unsecured MLE the initial byte has a value of 255, with no auxiliary header byte.

There are 18 generic MLE command datagrams available in *Thread*. These MLE commands address various aspects of establishing links, data requests, routing, and discovering neighboring nodes. These generic MLE commands can be sequenced to accomplish higher level tasks.

D. 6LoWPAN adaption layer

Thread uses IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) as specified in RFC 4944 [3] and

RFC 6282 [4]. 6LoWPAN allows use of IPv6 addresses for IoT devices within the PAN. These standards are used in *Thread* with very minor restrictions. This layer allows fragmentation and reassembling of IPv6 packets to and from IEEE 802.15.4 MAC data frames. 6LoWPAN is designed to carry datagram over highly constrained links, like IEEE 802.15.4, with limited bandwidth, small memory, low power and a transmission speed of only 250 kbps or less.

IEEE 802.15.4 specifies MTU (Maximum Transmission Unit) of 127 bytes, of which only 80 octets of actual security enabled MAC payload are available. 6LoWPAN allows a Mesh address header supporting sub-IP forwarding. By doing a header compression on IPv6 datagrams, 6LoWPAN reduces large IPv6 and UDP headers down to (in the best case) only a few bytes [4].

E. Network layer

Thread devices have to implement, IPv6 Specification, RFC 2460 [5]. However, *Thread* devices don't have to implement all the features, see Chapter 5 [23] for exceptions.

A *Thread* device must implement IPv6 addressing architecture, RFC 4291 [6]. *Thread* uses two scopes for addressing: link-local which is reachable in a single radio transmission and realm-local which is reachable within the PAN with hops. *Thread* devices must support one link-local and at least two realm-local addresses (used for communication inside the PAN). *Thread* devices may also support additional IPv6 address like, Unique Local Address (ULA) and Global Unique Address (GUA), if resources are available. *Thread* devices must also support link-local and Mesh-local multicast addresses.

Unicast IPv6 addresses are used for communication within the Mesh. Unicast can be Routing locator, Anycast locator (Leader, DHCPv6, Service, Commission, Neighbor discovery) or Endpoint Identifier (EID).

Thread networks do not depend on Dynamic Host Configuration Protocol v6 (DHCPv6) and *Thread* devices are not required to implement it.

Thread devices are required to implement Internet Control Message Protocol for IPv6 (ICMPv6), RFC 4443 [7], in its entirety with only minor modifications.

Routing protocol

Thread uses a simple distance vector routing protocol. Routing within PAN is based on RIPng, RFC 2080 and RFC 1058 standards. PAN routers periodically transmit their routing "costs" to all the other routers and their quality of one-hop links. Routing cost for message can be computed from this. Each router also maintains a routing database for each interface that uses distance routing. The database contains: router-id, link, and route. PAN routers keep track of the *Thread* version in their child link and proactively unicast newer versions. Additionally, if the router is a Leader it has a database of id-assignments. Leader router collects, collates, and distributes information about Border routers and other servers available to the *Thread* network via MLE. The Leader router also manages 6LoWPAN context ID assignments to IPv6.

Thread also has a concept of a full set and a stable set of network nodes. Since mobile devices in the PAN can drop off and rejoin, the devices have an option to only hold the stable set.

Thread network can partition into disjoint sets due to communication breakdown. Therefore, *Thread* includes a provision allowing each partition to operate as a separate network. Later on, if the separate partitions get reconnected they can merge back into a single network. A Border router or REED can start a new partition if the Leader node cannot be reached. End Devices, cannot create a new partition, but are always attached to their parent routers and belong to the same partition as their parent. The *Thread* network is self-healing in this regard.

F. Transport layer

The *Thread* Transport layer is based on User Datagram Protocol (UDP) described in RFC 768 [8] and Requirements for Internet Hosts, RFC 1122 [9].

Generally, IoT protocols use one of two main application messaging protocol at the Transport layer, Constrained Application Protocol (CoAP) [20] [30] and Message Queuing Telemetry Transport (MQTT) [21][31]. Both of these messaging protocols were specifically designed for low power IoT type devices. *Thread* uses CoAP, due to its low memory, and low processing requirements. MQTT due to its use of TLS is not well suited for *Thread*.
The actual transport protocol used by CoAP is a UDP*, which enforces use of* Datagram Transport Layer Security (*DTLS)*, IETF RFC 6347 [10].

CoAP has a set of security modes and mandatory-to-implement ciphers. CoAP borrows some concepts from Representational State Transfer (REST) protocol [22]. CoAP can be operated in "no security" mode, "pre-shared key" mode and "certificate" mode. CoAP**s** (CoAP secured) can also be configured in "raw public key" [IETF RFC 7250]. In pre-shared key mode CoAPs hashes pre-shared keys with a list of corresponding communication nodes. Although use of certificate mode is well established, its use in IoT is discouraged due to resource constraints.
.
A "no security" mode on CoAP can be used in *Thread* if the Data link layer encryption is available and being used.

Thread uses DTLS encryption for the Transport layer, if the radio hardware does not support data encryption. Encrypting the data twice, both at Data link layer and Transport layer is also possible. Double encryption can decrease the data compression ratio, leading to more package transmissions and higher power consumption.

G. Security and device commissioning

Thread uses Mesh Commissioning Protocol (MeshCoP) which allows for securely authenticating, commissioning, and joining new-untrusted radio devices to a mesh network.

Thread PAN is an autonomous self-configuring Mesh of devices with IEEE 802.15.4 interface and link level security layer. Each device in the PAN must possess a current-shared secret master key. To add a device to the PAN, a human administrator must authenticate it to the network as eligible for joining, followed by the commissioning of the device and supplying it with the master key for the PAN over a secured channel.

The Commissioner node is an authentication server, which is authorized to provide network credentials for new devices. The Commissioner can be internal or external to the PAN. This node can be a server in the Cloud or smart phone, which user uses for managing the joining of the device to the PAN.

Before a device can be authenticated through the Commissioner, the Commissioner itself must be authenticated. The Commissioner's authentication step establishes a secured client/server socket connection, called the Commissioning Session. This happens between the Commissioner and the Border Router (BR) via DTLS (RFC 6347 [10]) or TLS (RFC 5246 [11]). Commissioning Session uses an assigned UDP port as advertized during the discovery phase. The credential used in the Commissioning Session is known as Pre-Shared Key for Commissioner (PSKc). Once the authentication is successful the Commissioner registers with the Border router and the Border router conveys the Commissioner information to the Leader node. The Leader either accepts or rejects the Commissioner. If the Commissioner is accepted the Leader propagates the Commissioner's information to the Joiner Routers (JR) in the PAN.

A device requesting to join the PAN is called a Joiner Device (JD). For a JD to join the PAN, it must transmit Discovery Request/Response messages using MLE in unsecured mode to a JR. To be successful the JD should receive the Response message before the joining timer expires. JR sends a Discovery Response with joining UDP port, only if joining is enabled, otherwise no message will be sent.

JD will perform DTLS handshake to establish a Joiner Session with JR. The JRs will relay UDP messages to the BR, which in turn will relay them to Commissioner. JR and BR do not have access to the content of these messages and only relay the messages which are arriving at the assigned UDP port. JD and The Commissioner then exchanges the token to establish trust. Commissioner inspects JD Interface Identifier (IID) and credentials. If the Commissioner is satisfied with the responses from the JD, the JD is then provisioned with the appropriate data and services, and also provide Key Encryption Key (KEK) and the Joiner Session will be closed. The JR is then signaled by the Commissioner that JD is to be entrusted with network credentials. Then the Commissioner finally provides a shared secret between JD and JR, to complete the joining.

Each *Thread* node receives a Master Key when joining. This Master Key is used in conjunction with a sequence counter to derive two separate keys, one for MAC layer and other for DTLS. Using Hashed Message Authentication Mode with SHA-256 algorithm (HMAC-SHA256) produces a 32-bit output [RFC 6234]. The upper 16-bits are used for MAC key and the lower 16-bits are used for DTLS key. The keys are rotated based on one of these criteria: key index changes in the neighboring nodes, or the key rotation timer expires, or the incoming MLE matches the next key.

Almost all IoT protocols, including *Thread*, offer Advanced Encryption Standard (AES) 128-bit [13] [17]. However, AES does not scale well with hundreds of devices in the network and requires very large keys. Therefore, *Thread* [28] uses Elliptic Curve Cryptography (ECC), RFC 4492 [12] [18]. ECC is an asymmetric, public-key method that scales well and provides a higher level of security for the same number of bits. Other low power public key encryption algorithms like Rabin's Scheme and NtruEncrypt are also good candidates for IoT [19].

III. CONCLUSION

Thread Group Inc. started from a set of requirements to develop *Thread*. They developed a robust open standard which is vendor neutral. One distinct limitation of *Thread* is that it is not necessarily interoperable with other IoT frameworks like *AllJoyn*, *ZigBee*, *Z-Wave*, etc.

The layers defined by *Thread* are based on either an existing standard or a RFC. No new standards or RFCs have been proposed by *Thread*. So in terms of technology advancement nothing new is proposed, but in terms of standardization of IoT *Thread* can be a very important step forward. One of the critiques of current situation in IoT marketplace is that, IoT devices from different vendors and protocols don't work with each other. Each IoT protocol needs a separate hub and app to control it [24][25]. This is due to multiple propriety systems in the market today. The

Thread standard proposes to gather enough momentum to cause these propriety systems owners to merge under a unified platform.

In building this standard the *Thread* standard overcame several challenges. Today the *Thread* developers included a low power wireless network. As the IEEE 802.11a/g/n/ac radios are too power hungry and IEEE 802.11ah is not mature, but IEEE 802.15.4 was available and low power. IEEE 802.15.4 is interference tolerant and works in 2.4 GHz ISM band world-wide making it a good choice. Since time to market is always a concern, by choosing IEEE 802.15.4 a FCC certified radio became available. The readily available radio silicon saved time and avoided development cost. IEEE 802.15.4 radios can be software defined to be Bluetooth LE, Bluetooth, 802.15.4, 6LoWPAN, Gaussian Shift Key (GSK) or many other settings. Device vendors can buy the radio from a wide choice of manufacturers, Nordic semiconductor, Dialog semiconductor, Freescale, TI, Microchip, etc, for their use.

Thread developers also wanted a secure, self healing, user friendly, resilient topology operating without consumer intervention. The Mesh topology accomplished all of this. The Mesh topology can be re-route if a component fails, thus making it resilient. The Border Router is a single point of failure many other IoT frameworks, Thread Group solved this failure issue by allowing multiple Border Routers in their Mesh network.

Using 6LoWPAN adaption layer allows *Thread* to have an IP based network, with IPv6 addressing. The upper layers of protocol have been kept open to allow development by others.

Although, no consumer level product are available with *Thread* logo at the time of this writing, a presentation given by Thread Group Inc. claims that "a software upgrade can add *Thread* to currently shipping 802.15.4 products" [29].

One of Thread Group Inc. goals is to work with other IoT alliances to "reduce fragmentation" [26]. In this regard, *Thread* and *ZigBee* recently announced that *ZigBee* 3.0 application libraries will support *Thread* [27]. With the hope that this type of consolidation will continue and a single IoT protocol will emerge in near future.

IV. Future Work

Currently, we are working toward simulating the Thread protocol using NS2 simulator [32]. Thus enabling researchers to study and model the behavior of current, and alternate, features in *Thread.*

Disclaimer: At the time of writing of this paper, none of the authors were affiliated with Thread Group Inc.

References

[1] Aston, Kevin. "That 'Internet of Things' Thing", Available: http://www.rfidjournal.com/articles/view?4986, accessed on June 2, 2017.

[2] Gartner, Inc. "Gartner Says 8.4 Billion Connected 'Things' Will Be in Use in 2017, Up 31 Percent From 2016", http://www.gartner.com/newsroom/id/3598917, accessed on June 2, 2017.

[3] RFC 4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks, " https://tools.ietf.org/html/rfc4944

[4] RFC 6282, "Compression Format for IPv6 Datagrams over IEEE 02.15.4-Based Networks," https://tools.ietf.org/html/rfc6282

[5] RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification", https://www.ietf.org/rfc/rfc2460.txt

[6] RFC 4291, "IP Version 6 Addressing Architecture", https://www.ietf.org/rfc/rfc4291.txt

[7] RFC 4443, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", https://tools.ietf.org/html/rfc4443

[8] RFC 768, "User Datagram Protocol," https://tools.ietf.org/html/rfc768

[9] RFC 1122, "Requirements for Internet Hosts -- Communication Layers," https://tools.ietf.org/html/rfc1122

[10] RFC 6347, "Datagram Transport Layer Security Version 1.2," https://tools.ietf.org/html/rfc6347

[11] RFC 5246, "The Transport Layer Security (TLS) Protocol Version 1.2," https://tools.ietf.org/html/rfc5246

[12] RFC 4492, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," https://www.ietf.org/rfc/rfc4492.txt

[13] J. Daemen and V. Rijmen, AES Proposal: Rijndael, version 2, 1999. Available from URL: http://www.esat. kuleuven.ac.be/~rijmen/rijndael

[14] J. Chen, S. Kher, and A. Somani, "Distributed fault detection of wireless sensor networks," in *Proceedings of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks*, ser. DIWANS '06, 2006, pp. 65–72

[15] C. Lo, J. P. Lynch and M. Liu, "Distributed Reference-Free Fault Detection Method for Autonomous Wireless Sensor Networks," in *IEEE Sensors Journal*, vol. 13, no. 5, pp. 2009-2019, May 2013. doi: 10.1109/JSEN.2013.2244881

[16] I. Chatzigiannakis and A. Strikos, "A decentralized intrusion detection system for increasing security of wireless sensor networks," *2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007)*, Patras, 2007, pp. 1408-1411. doi: 10.1109/EFTA.2007.4416949

[17] Federal Information Processing Standards , "Announcing the ADVANCED ENCRYPTION STANDARD(AES)", Federal Information Processing Standards Publication 197 November 2001 [online] Available: http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[18] Z. Liu, X. Huang, Z. Hu, M. K. Khan, H. Seo and L. Zhou, "On Emerging Family of Elliptic Curves to Secure Internet of Things: ECC Comes of Age," in *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 3, pp. 237-248, May-June 1 2017. doi: 10.1109/TDSC.2016.2577022

[19] G. Gaubatz, J. P. Kaps, E. Ozturk and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, Kauai Island, HI, 2005, pp. 146-150. doi: 10.1109/PERCOMW.2005.76

[20] C. Bormann, A. P. Castellani and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," in *IEEE Internet Computing*, vol. 16, no. 2, pp. 62-67, March-April 2012. doi: 10.1109/MIC.2012.29

[21] "Oasis Message Queue Telemetry Transport", *OASIS MQTT version 3.1.1, 2014*

[22] S. Zamfir, T. Balan, I. Iliescu and F. Sandu, "A security analysis on standard IoT protocols," *2016 International Conference on Applied and Theoretical Electricity (ICATE)*, Craiova, 2016, pp. 1-6. doi: 10.1109/ICATE.2016.7754665

[23] Threadgroup Inc. website: http://threadgroup.org/What-is-Thread/Connected-Home

[24] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim and S. R. Chaudhry, "IoT architecture challenges and issues: Lack of standardization," *2016 Future Technologies Conference (FTC)*, San Francisco, CA, 2016, pp. 731-738. doi: 10.1109/FTC.2016.7821686

[25] S. S. I. Samuel, "A review of connectivity challenges in IoT-smart home," *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, Muscat, 2016, pp. 1-4. doi: 10.1109/ICBDSC.2016.7460395

[26] https://threadgroup.org/what-is-thread/consumers (Accessed June 7, 2017, see FAQ)

[27] "The Thread Group Expands Influence through Partnership with ZigBee Alliance, TCLA and Innovation Enabler Award", (Accessed on June 7, 2017) http://mysmahome.com/news/5905/the-thread-group-expands-influence-through-partnership-with-zigbee-alliance-tcla-and-innovation-enabler-award-2/

[28] S. Rajesh, "Ubiquitous energy efficient aquaculture management system," *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, 2016, pp. 1124-1128. doi: 10.1109/ICACCI.2016.7732195

[29] Thread: Introduction, Pg.3, (Accessed Jun 7, 2017) available at http://threadgroup.org/portals/0/documents/thread_introduction_website_7-15-14.pdf

[30] M. B. Tamboli and D. Dambawade, "Secure and efficient CoAP based authentication and access control for Internet of Things (IoT)," *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2016, pp. 1245-1250. doi: 10.1109/RTEICT.2016.7808031

[31] M. B. Yassein, M. Q. Shatnawi and D. Al-zoubi, "Application layer protocols for the Internet of Things: A survey," *2016 International Conference on Engineering & MIS (ICEMIS)*, Agadir, 2016, pp. 1-4. doi: 10.1109/ICEMIS.2016.7745303

[32] NS2 Simulator, documentation and download links can be found at (Accessed on January 20, 2018): https://www.isi.edu/nsnam/ns/