Security in the IoT Ecosystem

An overview of IoT operations and security concerns

Khalil D. Stemmler
Brock University
St. Catharines

Abstract—Security on the Internet is not an easy task for engineers and developers. With every security hole plugged and concern addressed by the “good guys”, the bad guys merely need to locate a single chink in the wall of defense to be given an opportunity for malice. In this society that places trust in technology, society is also placing trust in the engineers that create this technology. The IoT sphere is not exactly new but it’s not exactly old either; and still, it faces a number of particular concerns with respect to security. This document aims to address the shortcomings of security in the current IoT ecosystem and mechanisms that make the best of what’s currently available.

Index Terms—IoT, Security, Wireless Sensor Networks, IPv6

1 Introduction

THE Internet of Things refers to the interconnection of billions of "smart objects" gaining popularity today following a number of communication patterns such as: human-to-human (H2H), human-to-thing (H2T), thing-to-thing (T2T), or thing-to-things (T2Ts) [1].

In 1998, the IETF (Internet Engineering Task Force) started discussion about a new protocol to phase out IPv4 over the coming years. There was an explosion of sorts in the number and range of IP capable devices being released in the market and the usage of these by an increasingly tech savvy global population [2]. While IPv4 address exhaustion has been predicted since the 1980s, we’ve employed a variety of different IPv4 exhaustion mitigation techniques to sustain IPv4 temporarily [3]. In 2014, India was allocated somewhere near 35 million IPv4 addresses to use with a user base of about 360 million data users; an impossible situation only made possible through problematic techniques like NATing and subnetting layers and layers of subnets [4]. In 2003, the final Internet Draft (RFC 4294) was written on IPv6 specifying requirements for IPv6-enabled devices [5]. With the advent of IPv6’s address space and its 2^128 addresses, IPv6 has 1) allowed the ability for every device to obtain a uniquely identifiable IP address; 2) opened the door for IoT nodes and entire Wireless Sensor Networks to be able to homogeneously communicate with standard IP devices using upper layer internet protocols; 3) simplified the interface in which 'things' may talk to each other (T2T - thing to thing) [4].

Naturally, there have been some challenges with IoT. With respect to lifetime, constrained devices need to act efficiently when communicating with other ‘things’. It has been proven that 70% of energy consumption in constrained devices is caused during the data transmission process [6]. A number of upper layer protocols (HTTP/S, SSL/TLS, SMTP, SSH) rely on the Transmission Control Protocol (TCP), the connection-oriented transport protocol. TCP works well in low packet loss networks but suffers tragic inefficiencies in environments where packet loss is abundant. This is relevant because the factors of wireless sensor networks and the nodes that make them up suffer low processing power on ‘things’, and high chances of packet loss. Thus, lots of work has gone into establishing new and efficient UDP-based all-IP protocols that allow IoT devices and WSNs interoperability with standard IP systems.

With respect to security, a large number of consumer IoT devices today lack a strong security architecture. As we've seen on October 21, 2016 with the Dyn Attack in which hackers created a botnet of baby-monitors, IP-cameras and other IoT devices to perform a DDos on one of the world’s most widely used DNS providers, the balance of importance placed on security of these cheap devices vs. the ease-of-use and ease-of-configuration calls for changes in out of the box consumer IoT devices [7].

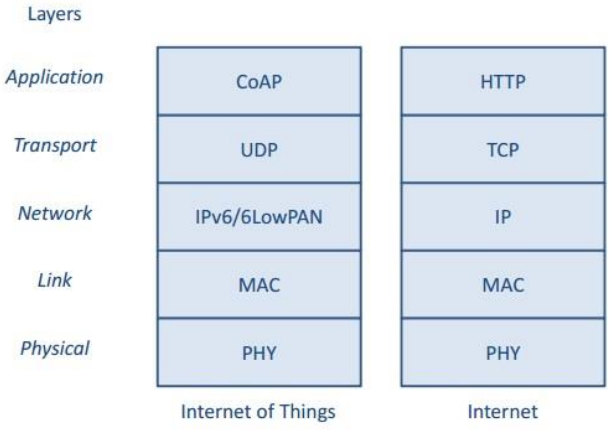

Figure 1 – Comparison of IoT and standard Internet OSI protocol stack.

One of the most common IoT technologies are RFID tags and readers. RFID is used for a number of applications such as tracking of goods, access management and tracking of persons and animals. In Ontario, Canada, riders in the GTA and Ottawa Area use the Presto card to board trains, buses and street cars across

multiple different transit companies [8]**.** These systems must realize that security is a critical issue that needs to be addressed correctly. Our discussion shows that there needs to be a standardization of constrained-device specific crypto-protocols to be set into place and a systematic approach for IoT security must be taken to reach certain security obligations for our products of tomorrow.

This paper presents an overview of the current IoT ecosystem, its operation and places a special focus on security and the work being done to better standardize secure all-IP IoT implementation.

2 Lifecycle of a Thing

We will now underline the lifecycle of an IoT *thing* (from now on, we will use the terms *thing* and *node* synonymously) and classify the important security aspects with respect to the lifecycle events.

2.1 Bootstrapping

The lifecycle of a thing begins the when it's manufactured. Take for example Building Automation Control Systems (BAC). These provide automatic control schemes for heating, ventilation, and air-conditioning (HVAC) systems, lighting, and shading of indoor environments [9]. We'll use the installation of such a system to illustrate the lifecycle of an IoT thing.

When a network of IoT things are commissioned and being installed, it's very unlikely that every single node in the network will be made by the same manufacturer; that is, we have a heterogeneous network. For these networks, it's important that there exists some common interface for which the nodes can communicate with each other. To address this interoperability, the IEEE 802.15 working group defined the 802.15.4 technical standard for low-rate, wireless personal area networks (LR-WPAN). Within the OSI stack, the IEEE 802.14.5 communication protocol defines Layers 1 (physical) and Layers 2 (MAC / Data Link layer), leaving the upper layer protocols unspecified but commonly aided through the use of ZigBee, MiWi, Thread and more interestingly to us, 6LoWPAN (the constrained-device IPv6 protocol). We talk more about upper layer protocols in **Section 5: IoT on the Internet**.

The most important feat about this lifecycle event is the creation of secret keys and the device identity for the newly commissioned and installed thing.

In summary, the bootstrap phase needs to outline:

- How a *thing* will obtain an identity
- How it will communicate own its identity to another thing (so that they may form a ***security domain*** before data transfer); e.g: see to the transfer of security parameters for trusted operation.

A *security domain* is a secure communication channel between two or more parties that is kept separate from other networks and ideally provides some of the basic security services such as confidentiality, authentication, integrity, authorization, non-repudiation and availability.

To meet these requirements, a security architecture for the network must be employed. The two types of security architectures are **distributed** architectures and **centralized** architectures.

- **Distributed** security architectures imply Diffie-Helman handshakes and ad-hoc forming of a security domain between two nodes to communicate with each other.
- **Centralized** architectures use preconfigured keys or certificates held by a thing for the distribution of operational keys in a given security domain. This architecture affords a much higher level of administrative authority over whom each node can talk to.

Most networks will employ both distributed and centralized architectures in some way. We talk more about this and explore best practices for how to address the requirements of the bootstrap phase in **Section 5.1: Distributed vs. Centralized Architecture**. It should also be noted that the Bootstrapping phase doesn't have to be one particular event. In fact, bootstrapping can be stretched out over a long period of time as more devices are commissioned as well.

2.2 Operation

Following the installation and commissioning of a thing, it runs the operation of the system (in our example, the BAC system). Sometimes, *things* need software updates or may need to be completely reconfigured; thus, the lifecycle of a thing is a cycle from the initial bootstrap, operation, updates/reconfiguration, operation again and so on until the thing is finally decommissioned. This lifecycle is generic and can be applied to most IoT scenarios.

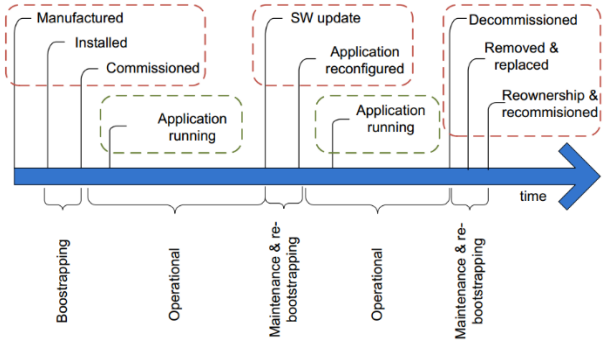

Figure 2 – The lifecycle of a thing in the IoT

Operational expenses can amount for systems that employ independent networks for each subsystem. For example, if the lighting subsystem and heating subsystems were on different networks, this increases complexity in terms of maintenance and configuration and often requires physical access to the things in the network. A trend is to employ all-IP based networks using 6LoWPAN by putting a gateway to translate IPv6 to IPv4 and vice versa so that network administrators may access the wireless sensor network of things remotely [10]. Connecting the IoT domain to the Internet introduces challenges addressed in **Section 5.4: Heterogeneous Communication**.

3 General IoT Architecture

Furthermore to our investigation of IoT security, we will aim to understand the general layers of an IoT implementation. In [11], the general listing of the layers of a *thing* is described as having a perception layer, network layer, middleware layer and an application layer. While an IoT thing can autonomously possess each and every one of these layers, we pay more attention to the IoT architecture in general as it pertains to an entire Wireless Sensor Network of nodes.

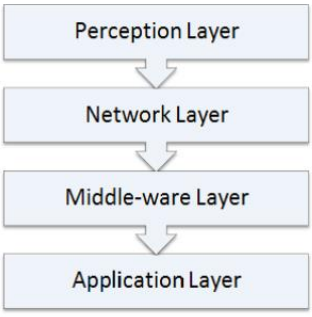

Figure 3 – IoT layered architecture.

3.1 Perception Layer

The perception layer consists of sensors or a Wireless Sensor Network that the *thing* can use to identify changes in its environment [12]**.** In Wireless Sensor Networks where the primary function is to relay information to some central repository, it can be easily understood why the perception layer sits underneath the Network Layer. For some IoT devices, the sensors are used primarily to update the *thing's* understanding of the world; in that case, making sense of perceived information is often done within the Middle-ware Layer instead- seemingly skipping the Network Layer.

More complex sensors such as RFIDs and Barcodes are basically used to uniquely identify objects. Sensors like these are subject to the requirement of better security because of the trust placed in the inherent uniqueness and integrity of the barcodes and RFIDs being sensed/scanned.

3.2 Network Layer

The network layer exists to transmit information that's been gathered from the perception layer, to other nodes in a Wireless Sensor Network or a processing system. Communication can take place over Internet, Mobile or any other kind of network [11, pp. 3].

3.3 Middle-ware Layer

The middle-ware layer is the most abstract layer. It can be a cloud-based gateway for accepting streams of data from *things* distributed throughout the real world. It can be a node that resides inside a Wireless Sensor Network that has been designated to perform computation on data received from the other nodes in the WSN.

In general for any node on the network, the middle-ware layer's purpose is to perform some sort of automated action based on the results of processed data received over the network from other nodes and/or from its own perception layer [11, pp. 3]. The middleware layer is usually connected to a database (locally or sometimes externally) to store perceptual information and the realizations of perceived info. The middle-ware layer can differ from node to node based on the service-type, e.g: the middle-ware layer inside of a plant-watering *thing* vs. a ventilation system will have vastly different **releasers**. A releaser is similar to that of a latch or a switch; denoting whether or not a stimulus or condition is met [13].

3.4 Application Layer

The application layer is the actual product. It can be a variety of practical applications spanning all kinds of industries; some common realms are Smart Home, Smart Environment, Smart Transportation and Smart Hospital, etc [11, pp. 3]. It can include a graphical user interface, the feature to remotely monitor the *thing* through a cloud-based system, etc. Information at this layer usually consists of user account credentials, profile information, settings, preferences and realizations made through crunching data at the middle-layer.

3.5 Security Concerns (at each layer)

In [1, pp. 4], the authors define the most important security aspects in the IoT with the following terminology:

Security Architecture:

This is comprised of all of the elements (either **physical** like gateways or **procedural** like protocols) that make up the security relationships between *things* and how interactions are handled (e.g., centralized or distributed).

Security model of a node:

Efforts such as storing keying materials securely, process separation, firewall rules on a single node defines how the security parameters, processes and applications are securely managed in a *thing.*

Security bootstrapping*:*

Through making use of the *security architecture*, this denotes the process by which a *thing* securely joins the IoT. Bootstrapping includes:

- Authentication of a device
- Authorization of a device
- Transfer of security parameters for trusted operation

Because this aspect makes use of the security architecture, it's another full IoT architecture security aspect.

Network security:

Describes the mechanisms applied within a network to ensure trusted operation of the IoT. Specifically, it prevents attackers from endangering or modifying the expected operation of networked *things*. Network security can include a number of mechanisms ranging from secure routing to data link layer and network layer security.

***Application security*:**

Guarantees that only trusted instances of an application running in the IoT can communicate with each other, while illegitimate instances cannot interfere [1, pp. 4].

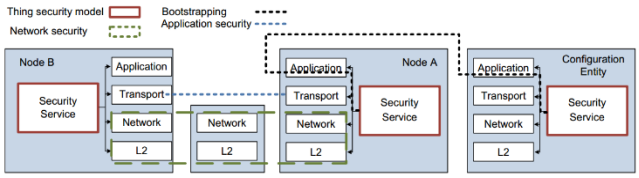

Figure 4 – Overview of IoT security mechanisms through relevant OSI model layers.

Next, let's discuss some of the most relevant security concerns as it pertains to each layer of the IoT architecture as suggested by [11].

3.5.1 Perception Layer Security Concerns / RFID Technologies

The perception layer employs a variety of different sensors for different application requirements; one of the most common of which are RFIDs.

Mentioned in the introduction, RFIDs are used for things like access control, tracking humans, products and animals, etc. With respect to RFID security, today, the main security threat is within front-end RF communication [14]. Standard IP-security protocols between RFID readers and the network separate this concern to the network layer but at the perceptual level, the real threat is between tags and readers [14].

For applications that require the use of RFID (Radio-frequency identification), we will aim to bring attention to some of their challenges.

1. *Unauthorized access to RFID tags*. "The RFID allows for the identification of objects and/or subjects remotely using attached RFID tags via a radio frequency channel, hence identification is achieved in a contactless manner" [15]. A large number of RFID systems lack proper authentication mechanisms; this allows tags to be accessed by third parties without authorization. The real danger here is that the attack cannot only read the data but the data saved on the tags can be modified or deleted by the malicious third party [16].

2. *Tag cloning.* Tags are deployed out in the wild on different objects and for a wide variety of contexts such as process tracking, key card access, etc. Because the cards are physically visible, they're vulnerable to physical hacking techniques if hackers are able to be capture them. Hackers in possession of stolen tags are able to create a replica of the tag and compromise the system by providing unauthorized access as the card reader will not be able to distinguish between the legitimate and cloned tag [17].

One device that is capable of ripping data from tags is the BLEKey (Bluetooth Low Energy Key) [18]. Installed onto an RFID reader, it can rip and store data from up to 1500 RFID cards. It was designed by two security researchers, Mark Baseggio from security firm Accuvant and *Eric Evenchick* from Faraday Future primarily to show the vulnerability of the *Wiegand communication protocol* which in 2015, was the most widely used RFID card readers [19]. The tool can also be combined with a long range reader to skim data from RFID tags from a distance.

3. *Eavesdropping.* Wireless networks, unlike wired ones, suffer from the physical confidentiality of data transfer because data travels through air. Without encryption, it becomes inherently easy for the hacker to sniff out any private information like passwords moving from tag-to-reader or reader-to-tag [20].

4. *Spoofing.* Spoofing is sending fake information from a malicious party, fooling the RFID system to assume that the sender is someone else on the system. Hackers use this to get full access to the system as a different user/trusted entity [21].

5. *RF Jamming.* Disrupting the operation of RF communication between tags and readers by blasting an excess amount of RF noise which spans the bandwidth that the RFID devices operate at.

6. *Tag Tracking.* Another concern is that RFID tags will be used by unknown third parties to track persons when they do not wish to be tracked.

3.5.2 Network layer Concerns

The network layer is concerned with the actual transmission of data in a Wireless Sensor Network. Security at this layer is a lot more matured than that of the perception layer but it still has its challenges nonetheless.

1. *Sybil Attack*. The Sybil attack, "named after the subject of the book *Sybil*, a case study of a woman diagnosed with dissociative identity disorder", is where a hacker can forge multiple and many identities from a single node and if desired, introduce a disproportionally large illegitimate influence over the system in terms of population [22].

2. *Sinkhole Attack.* In a Sinkhole attack, a hacker utilizes a compromised node on the network and advertises attractive routing updates to its neighbors using whatever routing protocol is running in the WSN. These attractive updates effectively tell the neighbors that they should send their traffic through the compromised node as it's the best-cost path. However, the compromised node doesn't actually forward traffic to the location that it's supposed to go to, instead it just drops packets. In a situation like this, all of the traffic on the network gets "sucked" into a "sinkhole" and the senders believe that their data has been received by the recipient [23].

3. *Sleep-Deprivation Attack.* One of the primary concerns for constrained-devices is battery power and the lifetime of a device on batteries. In a Wireless Sensor Network, these *things* expand the lifetime of their batteries by following battery-saving sleep routines. Sleep Deprivation is self-explanatory; these are attacks that look for ways to keep the IoT nodes in the network awake for as long as possible so that they may not last long and will shut down.

4. *Denial of Service (DoS).* This is an attack in which the hacker floods the network with garbage traffic in the attempts to

exhaust the resources of the targeted systems. The network becomes so flooded with traffic that it becomes unavailable for any legitimate traffic to make its way through it.

5. *Malicious Code Injection.* This type of attack has the potential to be the most serious attack on the network. It works through a compromised node on the network; if the hacker has the ability to inject malicious code into some part of the system, they could shut down the network entirely or elevate their remote access from one node to full control of the network.

6. *Man-in-the-Middle.* MITM attacks are a form of eavesdropping that trick two hosts on a network to believe that they are talking directly to each other over a private channel while in reality, they are actually sending their raw packets to an unauthorized party who forwards the packets to their targeted destination after stealing data from and potentially altering the packets. This is also precursor setup for a Denial of Service attack as well.

3.5.3 Middle-ware Layer Concerns

The middle-ware layer's primary concern is to keep the data that has been acquired from each of the nodes on the WSN secure. Here are some of the security challenges:

1. *Unauthorized Access.* For some of these middle-ware layer systems, there will exist interfaces and ways to manage the server and connect to databases to save perceptual data. Access to this system is of utmost importance and must be hardened; for if a hacker is able to gain unauthorized access to this system, they could potentially steal or destroy as much data as they wish.

2. *Denial of Service (DoS).* The DoS attack is similar to that of which happens in the network layer, but if we consider it here in the middle-ware layer, we can consider a specifically targeted attack to the gateway servers. One effective way that hackers can tie up the resources in a server is to perform a Slow Loris attack, filling up all network connections on a server with slow-running requests, disallowing legitimate connections to the server [24].

3. *Malicious Insider.* This occurs when someone with privileged access to the system decides they want to use their access "for personal benefits or the benefits of some 3rd party" [11].

3.5.4 Application Layer Concerns

Application security issues consist of issues that target either a) the user using the application or b) the backend through the application itself.

1. *Malicious Code Injection.* If a hacker is able to get into a user's computer or mobile device, they can leverage an attack through loading malicious code onto the device to secretly steal information from the user's application profile.

2. *Phishing.* Phishing is an email spoofing attack where the victim is fooled into opening an email and the hacker is given access to the credentials of the user. They can then use these credentials to gain access to more sensitive information about the user in the application.

3. *Sniffing.* "An attacker can force an attack on the system by introducing a sniffer application into the system, which could gain network information resulting in corruption of the system" [11, 25].

4 Security Goals

As per [11], at a high level, we now identify the security practices that we should maintain at each layer. These practices, when applied diligently in a system, can aid a system in obtaining the requirements for WSN operation: **confidentiality**, **authentication**, **integrity**, **availability** and **quality of service** [26].

a) Confidentiality: Privacy of sensitive information and restricting that data to only the users which are authorized allowed to view the data.

b) Authentication: Being able to claim an identity and verify it with non-repudiation.

c) Integrity: the data is not to be transformed on its way from sender to receiver.

d) Availability: services are needed and in demand.

e) Quality of Service: timely and accurate packet delivery.

4.1 Perception Layer Goals

As discussed in **Section *3.1. Perception Layer***, the Perception Layer is at the bottom of the IoT architecture and is mostly concerned with security mechanisms for the physical sensors that are distributed within the WSN.

Using the general *security toolkit* (see Figure 5) at this layer, we can attempt to combat malice and achieve the our Perception Layer goals which include: **Authentication**, **Data Privacy**, **Privacy of sensitive information** and **Risk Assessment**- each of which are discussed in this section.

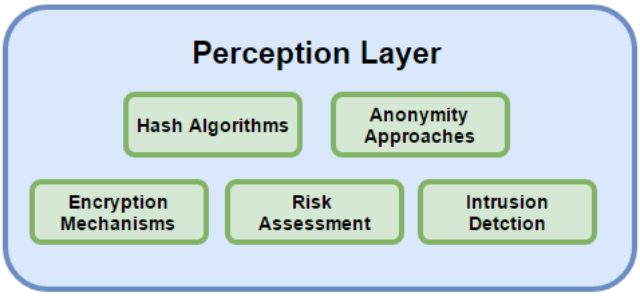

Figure 5 – Security toolkit for success at the perception layer.

1. *Authentication.* Authentication typically incorporates some form of cryptographic hash function like SHA-1, SHA-2 or MD5 to create a string from a secret key that only some entities on the system should know based on the security architecture. With PKI (Public Key Infrastructure), each node has a public and private key. Without PKI or a key management infrastructure, we can rely on key exchange protocols such as MIKEY and Diffie-Helman to establish shared secret keys between two parties to

use as inputs to create a hash on both sides and authenticate an accompanying message. These hash functions must be able to withstand known attacks like Brute force, Side-channel and Collisions, etc.

To combat the issue of unknown third parties tracking RFID tags, protocols like TLS and DTLS provide the option of only authenticating the responding host so that the initiating host can stay anonymous.

2. *Data Privacy.* Data Privacy is achieved through encryption. More specifically, after using symmetric key algorithms to establish shared secret keys, we can then use asymmetric algorithms like RSA, BLOWFISH, DES and DSA to prevent eavesdroppers to be able to collect and decrypt sensor data. For tag to reader, this can be easily implemented into the sensors with low power consumption [11].

3. *Privacy of Sensitive Information.* There exists an approach called **K-Anonymity**. The concept of this approach is that we can anonymize columns with potentially sensitive data (such as identity, location and age) to make it harder for malicious parties to arrive at conclusions of the data if stolen. For example, in Figure 6, we see a non-anonymized table of patient records.

Name	Age	Gender	State of domicile	Religion	Disease
Ramsha	29	Female	Tamil Nadu	Hindu	Cancer
Yadu	24	Female	Kerala	Hindu	Viral infection
Salima	28	Female	Tamil Nadu	Muslim	TB
sunny	27	Male	Karnataka	Parsi	No illness
Joan	24	Female	Kerala	Christian	Heart-related
Bahuksana	23	Male	Karnataka	Buddhist	TB
Rambha	19	Male	Kerala	Hindu	Cancer
Kishor	29	Male	Karnataka	Hindu	Heart-related
Johnson	17	Male	Kerala	Christian	Heart-related
John	19	Male	Kerala	Christian	Viral infection

Figure 6 – Non-anonymized data table of 3 columns.

By using **suppression** (replacing all or some values of a column with "*") and **generalization** (replacing individual values with a broader range category), we can achieve k-anonymity as shown in Figure 7.

Name	Age	Gender	State of domicile	Religion	Disease
*	20 < Age ≤ 30	Female	Tamil Nadu	*	Cancer
*	20 < Age ≤ 30	Female	Kerala	*	Viral infection
*	20 < Age ≤ 30	Female	Tamil Nadu	*	TB
*	20 < Age ≤ 30	Male	Karnataka	*	No illness
*	20 < Age ≤ 30	Female	Kerala	*	Heart-related
*	20 < Age ≤ 30	Male	Karnataka	*	TB
*	Age ≤ 20	Male	Kerala	*	Cancer
*	20 < Age ≤ 30	Male	Karnataka	*	Heart-related
*	Age ≤ 20	Male	Kerala	*	Heart-related
*	Age ≤ 20	Male	Kerala	*	Viral infection

Figure 7 – Suppressed and generalized table with k-anonymity.

We say that this table has 2-anonymity because for "Age", "Gender" and "State of Domicile", there are at least 2 rows with the same tuple attributes [27].

4. *Risk Assessment.* Risk assessment is a part of Computer security that continuously discovers new threats to the computer system. DRAMIA (Dynamic Risk Assessment Method for IoT inspired by AIS) is one dynamic risk assessment method that works through analyzing network packets that float between the perception layer and the application layer in the IoT and computing a security risk value to decide whether a new threat has been detected [28].

5. *Intrusion Detection.* In addition to all of these security mechanisms, most RFID readers have an automated "kill" which can get sent to the RFID tag in the case of detecting an intruder. This prevents the RFID tag from being able to interface with the system [29].

4.2 Network Layer Goals

The network layer suffers from both wireless and wired types of attacks and because of the inherent ease of access to the wireless medium, conversations between nodes on the network can be easily monitored by a malicious party. Through the use of **P2P Encryption**, **Routing Security**, **Data Integrity** and **Intrusion Detection** (Figure 5), we can achieve our goals at this layer: **authentication**, **routing security** & **availability**, **data privacy**.

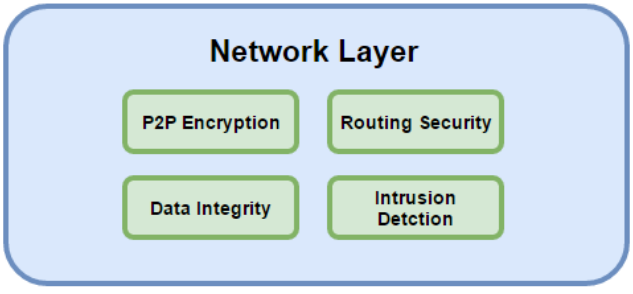

Figure 8 – Security toolkit for success at the network layer.

1. *Authentication.* Quite so often in IoT security architectures, there will be a key management server which can be used to establish trusted entities on the network. Through some sort of authentication process and through implementing **point-to-point encryption** between IoT devices and network elements, it becomes difficult for a hacker to gain access to any of the nodes within the sensor network.

2. *Routing Security & Availability.* When the network has been configured and trusted entities have been established through the authentication process, routing protocols can be configured on top of the network. There are a few routing protocols in particular that aim to ensure privacy of data exchange between the nodes and the processing systems in the network [11]. One of the routing algorithms is called **Source Routing** which is different from the typical form of routing; most routing protocols route packets across the network based on the destination address in the IP header, while Source Routing consults the source IP address. This differentiation ensures that not just any node of the

network's packets will be forwarded to a particular destination, only trusted nodes will have the privilege of having their packets forwarded by intermediate nodes on the network to get to the processing system.

These routing protocols are also meant to provide availability. In Wireless Sensor Networks, nodes use each other to forward traffic to across the network. When an intermediate node on the path to a particular destination is downed, the network implements control layer functionality to sense for changes to the network and re-converge with new routing rules based for each reachable destination. Multiple paths are also configured by default to improve the efficiency of routing data so that in case of a route failure, the node can immediately choose a new and feasible successor route. Functionality such as this aids in the **quality of service** for WSNs and aims to timely packet delivery.

3. Data Privacy. In addition to network monitoring mechanisms that monitor for intrusion on the network, data integrity methods involving cryptographic hash functions and secret keys are used to ensure that data received on the other end of a communication channel has been transferred unmolested by any third party or characteristics of the physical medium (satisfying quality of service's accurate packet delivery).

4.3 Middle-ware and Application Layer Goals

The middle-ware and application layer goals can be explored in unison similar to that of a **client-server** application system. The two layers are so closely coupled in each other's affairs that an integrated security mechanism is often required. Using the toolkit of **Authentication** mechanisms**, Encryption, Firewalls, Risk Assessment** and **Intrusion Detection**, we're able to achieve our goals between the two layers: **Authentication**, **Intrusion Detection**, **Risk Assessment** and **Data Security**.

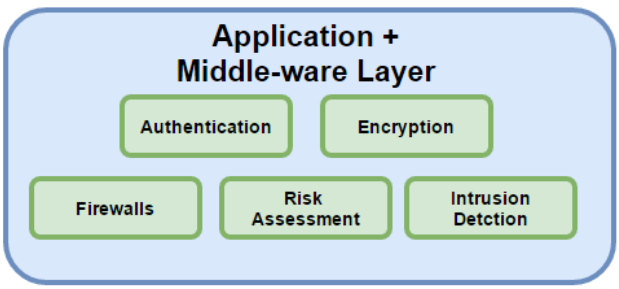

Figure 9 – Security toolkit for success at the application and middle ware layers.

1. Authentication. Authentication at this layer is done through some sort of process where the user gets access through some integrated identity broker systems. Examples of such systems are OAuth (which are commonly used by Google, Facebook, Microsoft and Twitter), LDAP, SAML, etc. This is similar to that of the other layers except that the user often has an option of what type of authentication scheme they would like to use and which associated information should be shared with the services.

Another area of utmost concern at this layer are the major technologies that are being used to power it: Cloud computing and Virtualization. It's quite common that parts of IoT topologies are virtualized within the cloud. Cloud Computing and Virtualization each introduce a plethora of security concerns, issues and best practices. Topics like horizontal VLAN hopping, policies, access to physical resources and more in Cloud Computing and Virtualization are beyond the scope of this document. Research is required in these domains to be able to promote a secure environment.

2. Intrusion Detection. At the application layer, intrusion detection techniques are plenty. Similar to the actions one would take in detecting intrusions on a web server, security threats can be alerted by generating an alarm in the occurrence of suspicious activity in the system. Through continuous monitoring of requests and logging the activities of users on the system, there are techniques to use this information to accurately generate alarms [30].

3. Risk Assessment. IoT engineers can use risk assessment to give justification for security strategies to harden the system.

4. Data Security. Through various encryption techniques, theft of data can be mitigated. Additionally, to prevent other malicious activities, Anti-Dos firewalls and up to date spywares and malware detection software can be introduced.

5 IoT on the Internet

In this final section, we discuss several different security architectures, common practices and suggestions for key management schemes and secure communication, interoperability between IoT and the Internet and constraints that can result in losses of efficiency.

5.1 Distributed vs. Centralized Architecture

We briefly mentioned distributed and centralized architecture earlier in **Section 2.1.** Bootstrapping; we continue the conversation of their operation and bootstrapping as in detail here. There are times when IoT devices need to do both distributed and centralized communication.

5.1.1 Distributed Operation

Distributed operation means that communication from thing-to-thing happens on an ad-hoc basis and the two nodes will need to form a security domain within them to perform secure communication. Not only will nodes communicate with each other, they may also call on backend servers and perform API calls on other systems on the internet [1 pp. 10].

5.1.2 Distributed Bootstrapping

The Diffie-Helman approach can be employed to allow for ad-hoc hosts to agree on a common secret. After that, IKEv2, HIP, DTLS and TLS can setup security associations and do key exchanges without needing to be connected to a certificate trust center.

An option is to use certificates and chains to determine authorized hosts with whom things will communicate with. This is fine but it can become cumbersome. Consider the scenario where we need to join several different administrative entities together; this means we would need some sort of common certificate anchor to achieve interoperability between the two separate domains [1 pp. 11].

5.1.3 Centralized Operation

Centralized architectures implement some form of a centralized security management server which is responsible for setting up group communication and establishing a security domain between IoTs nodes. While they can be used for backup of crypto keys, this design implies a single point of failure and it also implies that any time two devices want to communicate with each other, they need to have connectivity to the central security server. This makes it non-trivial to create ad-hoc security domains which is something that the distributed architecture is exceptionally well suited for. Ad-hoc communication in centralized architectures aren't impossible, they can actually be added later to the design of the system.

Centralized architectures are promoted in:

1. The ZigBee standard.
   a. ZigBee requests that there is some central trust center.
2. 6LoWPAN/CoRE working group.
   a. Proposals being done for 6LoWPAN/CoRE which aim to demarcate 6LoWPAN Border Routers (6LBRs) as devices which would be placed in the center of the security architecture.
3. A number of other internet protocols
   a. MIKEY (Multimedia Internet KEYing) is a key management protocol that is usually used for setting up encryption keys for a variety of real-time applications; usually multicast and group broadcast applications [1 pp. 10].

5.1.4 Centralized Bootstrapping

If an IoT node wants to join a network that it should have access to join, it can be preconfigured with keys or certificates and can be used in order to obtain operational keys from the central security server. For example, in a 6LoWPAN environment, PANA (Protocol for Authenticating Network Access) can be used with EAP (Extensible Authentication Protocol) to authenticate a PANA Client (PaC; the joining node) with a PANA Authentication Agent (PAA), which in this scenario would be the 6LBR. EAP, the authentication framework containing the parameters proving the identity of the IoT thing, is simply carried by PANA to the PAA and after successful authentication, the joining thing is provided access to the network [1, pp. 12].

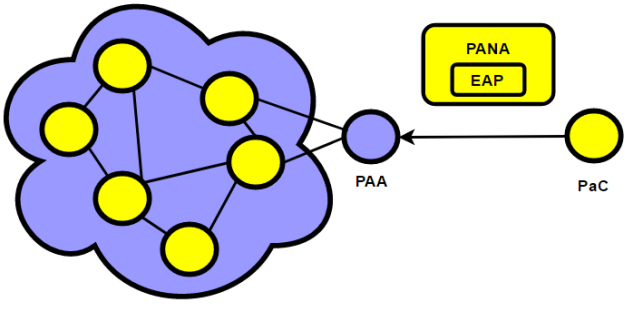

Figure 10 – Operation of PANA with EAP to authenticate a thing into a WSN.

5.2 Achieving Data Intregrity

We will now explain how MACs (Message Authentication Code) and other methods are currently deployed to achieve authentication through data transfer in IoT networks.

5.2.1 MAC

A secret Message Authentication Code (MAC) is piece of info to authenticate a message from one stated sender. It is sent alongside the encrypted or not encrypted message. It provides both **authentication** and **integrity** that the message has not been changed. MACs however, are different from digital signature because MACs are both generated and verified using the same secret key.

How does it work? Verifiers who possess the MAC's secret key are able to detect changes to the message content.
Who are the verifiers? Who has the secret key? It depends. The system could be implemented LIKE private-key cryptography where the sender uses their private key to digitally sign the message; and in this case, we would have perfect non-repudiation because it is the sender only who signs the message that they are sending and only the sender knows their private key.

More often than not, we'll be using a **network-wide** *shared secret-key* in which anyone in the network who is able to GENERATE a MAC can also VERIFY a MAC because both rely on using the same shared key. All authenticated nodes in the network should have access to the shared secret key (this keying material would be communicated through the bootstrap process). Thus, this leaves it impossible for any node on the network to perform non-repudiation of what they've sent because all nodes share the same key; there is no uniqueness. Each node is only able to determine if the message was or was not tampered with. This leaves much to be desired.

5.2.2 HMAC

By incorporating a cryptographic hash function like SHA-1 or MD5 in addition to a secret key, between two nodes we can generate a special type of MAC called a HMAC. The difference is little other than the MAC is created using a strong crypto hash function. We still face one problem; in fact- the same problem with regular MACs, establishing a common secret between two nodes over an insecure medium for secure communication. Note: a common secret is one of the ingredients required to create a MAC in the first place.

5.2.3 MIKEY-DHHMAC (HMAC-Authenticated Diffie-Helman for Multimedia Internet KEYing (MIKEY)

MIKEY (Multimedia Internet KEYing) is a key management protocol that is usually used for setting up encryption keys for a variety of real-time applications. One of the methods it employs to do this is called the MIKEY-DHHMAC (HMAC-Authenticated Diffie-Helman) exchange method which incorporates the Diffie-Helman Key exchange.

Diffie-Hellman is the de-facto key exchange protocol for establishing a shared secret over an insecure medium. While being a little more expensive in computation, DH does have **perfect forward secrecy** and can be used without any **PKI (Private Key Infrastructure)**.

The MIKEY-DHHMAC however, uses DH in a lightweight

way. Instead of computing RSA signatures like the standard MIKEY-DH exchange normally does (which is expensive), it works by using the HMAC to authenticate the two parts to each other, after having established shared sessions keys through DH.

Note that the MIKEY-DHHMAC key transport and exchange method is just one of 8 different methods that exist in the MIKEY key management protocol to setup a common secret to be used as a **session key**.

Even beyond MIKEY, there are other alternatives to integrating the key exchange process such as **IPSEC/IKEv2**.

5.3 Key Management Schemes

The following is a quick summary from [26, pp. 4] of some of the major cryptographic key management schemes that are being employed in the IoT for secure data transfer.

5.3.1 Network Wide Shared Key

This is by far the simplest method for key distribution which works by incorporating a single well-known key in the network. To communicate with another node in the network, a MAC is generated using the network wide shared key. The disadvantage is that if an unauthenticated user was to gain access to the network wide shared key, they could communicate with everyone on the network.

5.3.2 Master Key and Link Key

Relies on distributing a master to all nodes in the network beforehand and then using link keys for communicating between nodes. Primary disadvantage occurs when new nodes join the network; there isn't a secure way to transmit the link keys to the new hosts.

5.3.3 Public Key Cryptography

Implements a public/private key infrastructure to encrypt data and communicate with hosts.

5.3.4 Symmetric Keys

Every node has a set of link keys in advance for communicating with neighboring nodes. The disadvantage is that key management can become messy and non-scalable as there are (n-1)/2 keys for n nodes in the network.

5.3.5 Bootstrapping Keys

On-demand key generation done manually. Relies on a single point of failure and needs to keep a database of link keys for nodes allowed to communicate within the network.

5.3.6 Summary

Overall, it has been found that public-key cryptography is the most efficient and reliable scheme for Wireless Sensor Networks on account of their low memory usage, low CPU consumption, shorter key size, reliability through silent variable key management generation techniques [26, pp. 5].

5.4 Heterogeneous Communication

The IoT is tasked with the challenge of combining the resource constrained networks of the IoT with the daunting immensity of the Internet. Efficient and secure heterogeneity between these two networks is a working progress today.

5.4.1 Protocol Translation

6LoWPAN is a new version of the IPv6 protocol and the Low-power Wireless Personal Area Network (LoWPAN) that allows for constrained devices to participate in the IoT. It's primarily used in Wireless Sensor Networks behind an

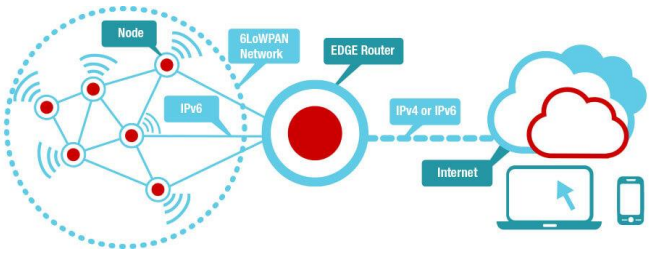

Figure 11 – Overview of 6LoWPAN in the Internet architecture (source: Texas Instruments)

edge router dividing the Internet from the IoT network.

The *Edge Router* handles protocol translation from 6LoWPAN's compressed IPv6 headers to either IPv6 or IPv4 in the core of the Internet.

CoAP (Constrained Application Protocol) is a new protocol specifically for constrained devices that is designed to translate easily to HTTP for a constrained version of the widely used application layer protocol.

The pattern we're currently seeing is that because of the subtle differences between the constrained protocols and their original Internet protocol that they're based on, gateways, bridges and protocol translators will become major obstacles for security measures. It is these devices that need a work to continue to be done on.

The two major concerns of gateways in security contexts are:

1) The relevant information to be translated is encrypted and inaccessible to the gateway.
2) Parts of the packet are integrity protected and rewriting would invalidate the packet.

Solutions to this include:

1) Sharing symmetric keys with gateway.
2) Reusing the Internet wire format in the IoT.
3) Selectively protecting vital and immutable packet parts.
4) Message authentication codes that sustain transformation.

None of these solutions are optimal for use between IoT and Internet either way as they sacrifice performance and security.

With respect to end-to-end authentication, as it currently stands, PANA with EAP provide the best functionality for authentication between IoT and Internet because of its layered approach [1, pp.10].

5.5 Wire-visible Constraints

In this section, we draw attention to some of the physical constraints that are faced by IoT nodes in a network; generally speaking, these constraints are unavoidable and are a limitation of the physical medium. The list outlines the important factors for light-weight implementations of the internet protocol suite in IoT [31, pp. 30].

- Checksum
- MTU
- Fragmentation and reassembly
- Options -- implications of leaving some out
- Simplified TCP optimized for LLNs
- Out-of-order packets

5.5.1 Checksum

In networking, the checksum is used to protect the IP-headers in a data transmission from data corruption. Checksum calculation occurs at each hop for a packet travelling through the network. This small computation is a necessary constraint to packet latency.

5.5.2 MTU

MTU (Maximum Transmission Unit) dictates the maximum size for an IP packet travelling through the network. In IPv4 default MTU size is 1500 bytes but in 6LoWPAN which utilizes IPv6, the MTU must be at least a 1280 octet. This constraint is relevant because it dictates exactly how much data can be sent in a single packet.

5.5.3 Fragmentation and Reassembly

Related in domain importance to MTU, fragmentation is something that is much scorned by network engineers. Fragmentation and reassembly occurs when a packet needs to pass through a node that has a MTU size that is smaller than that of the packet. Fragmentation slows down traffic on the network and reassembly of the packet must occur at the receiver.
Some security protocols like IPSec and IKEv2 do not allow packet fragmentation and reassembly and measures can be taken to avoid fragmentation in general [32].

5.5.4 Options – implications of leaving some out

IP Header options increase the size of the packets. With respect to MTU and fragmentation, sometimes IP header options will need to be omitted.

5.5.5 Simplified TCP optimized for LLNs

Low-Power and Lossy Networks (LLNs) are a classification of networks in which the routers and their interconnection mediums are constrained; for example, Wireless Sensor Networks. This consideration suggests a modified version of the TCP protocol which greatly simplifies its operation by stripping out some of the computationally expensive modules for efficient data flow. There are other people doing work to address this problem as well. RFC 6650 is a proposed standard for RPL, a Routing Protocol for Low-Power and Lossy Networks [33].

5.5.6 Out-of-order packets

Out of order packets is another common occurrence on IP networks. With TCP, it implements a *sliding window* algorithm that asks for the sender to retransmit any packets that were deemed dropped or out of order. While not a huge concern for more able-powered systems, this overhead can be additive to inefficient battery consumption on constrained devices.

5.6 Wire-invisible Constraints

Of equal importance to Wire-Visible constraints, wire-invisible constraints address some of the logical constraints that are faced by IoT nodes in a network. These are those that are influenced by the protocol design choices. The list outlines the important factors for light-weight implementations of the internet protocol suite in IoT [31, pp. 31].

- Buffering
- Memory management
- Timers
- Energy efficiency
- APIs
- Table sizes (somewhat wire-visible)

5.6.1 Buffering

Buffering of packets before routing them can be the bottleneck at high-traffic nodes on the network; this results in latency.

5.6.2 Memory Management

Memory management in IoT is as important as it is in any other computer system.

5.6.3 Timers

Routing protocols implement timers to keep the network converged. For example: in OSPF, a link state routing protocol, neighboring routers must receive a Hello Packet from its neighbor at a 10 second interval; 4 missed intervals implies that the neighbor is down and the network needs to re-converge.
This behaviour adds to the computation placed on an IoT node which may be inefficient with large networks with many neighbors.

5.6.4 Energy Efficiency

Each IoT node needs to employ the most efficient protocols and algorithms to prolong lifespan.

5.6.5 APIs

External APIs can be utilized; however, it's important to be aware of the efficiency of the API code base as it can work towards decreasing the lifespan of an IoT battery.

5.6.6 Table Sizes

Table sizes increase as the number of routes in a network increase. Consider what would happen if an IoT node accepted redistributed routes from a BGP router with over 1 million routes from the core of the internet.

5.7 Group Membership and Security

Up until this point, we've only discussed key establishment for unicast communication. With respect to T2Ts and Ts2T communication, group symmetric key establishment is another area of concern on the IoT for broadcast and multicast communication.

The versatile Diffie-Helman keys used in IKEv2 and HIP to establish symmetric keys for unicast communication could also be used for group symmetric keys. The question becomes, at which layer of the protocol stack should group symmetric keys be created?

Creating the group symmetric keys at the network layer using protocols like IPSec/IKEv2 or HIP/Diet HIP will be more beneficial in the long run than establishing the security domain through an application layer security protocol like TLS/DTLS. The reason that doing this at the network layer is better is because if we were to create group symmetric keys at the application layer, this would introduce the need for redundant security measures per application that wants to participate in group communication vs. the ability to do form a succinct group security domain at the network layer.

Mentioned earlier in **5.2.3**, the MIKEY architecture is a good choice for group key establishment for real time multimedia applications that rely heavily on broadcast and multicast communication within centralized architectures.

6 Conclusion

In this document, we've explored at a high level the ecosystem of the IoT and the security considerations and constraints that it must work to overcome at each layer of the IoT. We've discussed the lifecycle of an IoT thing and addressed the various architectures for bootstrapping a securirty domain in addition to the various architectures for IoT operation and measures that can be taken to ensure data integrity and secure communication for broadcast, multicast and unicast applications. We've explored some of the new constrained protocols that aim to achieve better performance gains from the constrained devices and problems with heterogenous communication between the IoT and the Internet.

References

[1] T. Heer, O. Garcia-Morchon, R. Hummen, S. Keoh, 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.

[2] K. Das, "IPv6.com - IPv6 History and Timeline", Ipv6.com, 2017. [Online]. Available: http://ipv6.com/articles/general/timeline-of-ipv6.htm. [Accessed: 22- Apr- 2017].

[3] R. Cannon, "Potential Impacts on Communications from IPv4 Exhaustion & IPv6 Transition", SSRN Electronic Journal.

[4] A. Taha Zamani, J. Ahmad, "IPv6 Adoption: Challenges & Security".

[5] "IPv6 Node Requirements RFC 4294-bis", Tools.ietf.org, 2017. [Online]. Available: https://tools.ietf.org/id/draft-ietf-6man-node-req-bis-02.html. [Accessed: 22- Apr- 2017].

[6] G. J. Pottle and W. J. Kaiseri, "Embedding the internet : wireless integrated network sensors," Communications of the ACM, vol. 43, no. 5, May 2000, pp. 51-58

[7] D. Attack, "Dyn Statement on 10/21/2016 DDoS Attack | Dyn Blog", Dyn.com, 2017. [Online]. Available: http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack/. [Accessed: 22- Apr- 2017].

[8] "Radio-frequency identification", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Radio-frequency_identification. [Accessed: 22- Apr- 2017].

[9] W.Kastner, G. Neugschwandtner, S. Soucek, and H. Michael Newman, "Communcation Systems for Building Automation and Control".

[10] "6lowpan Status Pages", Tools.ietf.org, 2017. [Online]. Available: http://tools.ietf.org/wg/6lowpan/. [Accessed: 22- Apr- 2017].

[11] M. U.Farooq, M. Waseem, A. Khairi and S. Mazhar, "A Critical Analysis on the Security Concerns of Internet of Things (IoT)", International Journal of Computer Applications, vol. 111, no. 7, pp. 1-6, 2015.

[12] Ying Zhang, Technology Framework of the Internet of Things and Its Application, in Electrical and Control Engineering (ICECE), pp. 4109-4112

[13] R. Murphy, Introduction to AI robotics, 1st ed. India, II: Prentice Hall of India, 2009.

[14] "RFID Security issues - Generation2 Security - ThingMagic.com", Thingmagic.com, 2017. [Online]. Available: http://www.thingmagic.com/index.php/rfid-security-issues. [Accessed: 22- Apr- 2017].

[15] Christy Chatmon, Tri van Le and Mike Burmester, "Secure Anonymous RFID Authentication Protocols".

[16] Mr. Ravi Uttarkar and Prof. Raj Kulkarni, "Internet of Things: Architecture and Security," in International Journal of Computer Application, Volume 3, Issue 4, 2014.

[17] Mike Burmester and Breno de Medeiros, "RFID Security: Attacks, Countermeasures and Challenges."

[18] "linklayer/BLEKey", GitHub, 2017. [Online]. Available: https://github.com/linklayer/BLEKey. [Accessed: 22- Apr- 2017].

[19] S. Khandelwal, "This $10 Device Can Clone RFID-equipped Access Cards Easily", The Hacker News, 2017. [Online]. Available: http://thehackernews.com/2015/07/hacking-rfid-access-card.html. [Accessed: 22- Apr- 2017].

[20] Benjamin Khoo, "RFID as an Enabler of the Internet of Things: Issues of Security and Privacy," in IEEE InternationalConferences on Internet of Things, and Cyber, Physical and Social Computing, 2011

[21] Aikaterini Mitrokotsa, Melanie R. Rieback and Andrew S. Tanenbaum, "Classification of RFID Attacks."

[22] "Real 'Sybil' Admits Multiple Personalities Were Fake", NPR.org, 2017. [Online]. Available: http://www.npr.org/2011/10/20/141514464/real-sybil-admits-multiple-personalities-were-fake. [Accessed: 23- Apr- 2017].

[23] George W. Kibirige, Camilius Sanga, "A Survey on Detection of Sinkhole Attack in Wireless Sensor Network".

[24] "gkbrk/slowloris", GitHub, 2017. [Online]. Available: https://github.com/gkbrk/slowloris. [Accessed: 22- Apr- 2017].

[25] Bhupendra Singh Thakur, Sapna Chaudhary, "Content Sniffing Attack Detection in Client and Server Side: A Sur-

vey," in International Journal of Advanced Computer Research, Volume 3, Number 2, 2013.
[26] H. Dogra and J. Kohli, "Secure Data Transmission using Cryptog-raphy Techniques in Wireless Sensor Networks: A Survey", Indian Journal of Science and Technology, vol. 9, no. 47, 2016.
[27] K.E. Emam, F.K. Dankar, Protecting Privacy Using kAnonymity, in Journal of the American Medical Informatics Association, Volume 15, Number 5, 2008.
[28] C. Liu, Y. Zhang, J. Zeng, L. Peng, R. Chen, Research on Dynamical Security Risk Assessment for the Internet of Things inspired by immunology, in Eighth International Conference on Natural Computation (ICNC), 2012.
[29] T. Karygiannis, B. Eydt, G. Barber, L. Bunn, T. Phillips, Guidelines for Securing Radio Frequency Identification (RFID) Systems, in Recommendations of National Institute of Standards and Technology.
[30] Animesh Patcha, Jung-Min Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, in Computer Networks, Volume 51, Issue 2, 2007.
[31] "draft-bormann-lwig-guidance-01 - Guidance for Light-Weight Implementations of the Internet Protocol Suite", Tools.ietf.org, 2017. [Online]. Available: https://tools.ietf.org/html/draft-bormann-lwig-guidance-01#section-6. [Accessed: 22- Apr- 2017].
[32] "RFC 7383 - Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", Tools.ietf.org, 2017. [Online]. Available: https://tools.ietf.org/html/rfc7383. [Accessed: 22- Apr- 2017].
[33] "RFC 7383 - Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", Tools.ietf.org, 2017. [Online]. Available: https://tools.ietf.org/html/rfc7383. [Accessed: 22- Apr- 2017].