

Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 2, 2017

O. Garcia-Morchon
Philips IP&S
S. Kumar
Philips Research
M. Sethi
Ericsson
March 31, 2017

**State of the Art and Challenges for the Internet of Things
draft-irtf-t2trg-iot-seccons-02**

Abstract

The Internet of Things concept refers to the usage of standard Internet protocols to allow for human-to-thing or thing-to-thing communication. The security needs are well-recognized and many standardization steps have been taken, for example, specification of CoAP over DTLS. However, security challenges still exist and there are some use cases that lack a suitable solution. This document first provides an overview of security architecture, its deployment model, security needs in the context of the lifecycle of a thing, as well as the state of the art on IoT security. Then, we discuss the concept of security profiles for the successful roll-out of secure IoT applications and describe remaining security challenges in the IoT.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 2, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

- [1. Conventions and Terminology Used in this Document](#) [3](#)
- [2. Introduction](#) [3](#)
- [3. Motivation and background](#) [4](#)
 - [3.1. The Thing Lifecycle](#) [4](#)
 - [3.2. Security building blocks](#) [6](#)
- [4. Managing Threats and Risks](#) [9](#)
- [5. State of the Art](#) [14](#)
 - [5.1. IP-based IoT Protocols and Standards](#) [14](#)
 - [5.2. Existing IP-based Security Protocols and Solutions . . .](#) [16](#)
 - [5.3. IoT Security Guidelines](#) [19](#)
 - [5.4. Guidelines and IoT Security Regulations](#) [21](#)
- [6. IoT Security Profiles](#) [22](#)
 - [6.1. Profiles Security Profiles](#) [24](#)
 - [6.2. Security Architecture](#) [26](#)
 - [6.3. Security Model](#) [28](#)
 - [6.4. Security Bootstrapping and Management](#) [29](#)
 - [6.5. Network Security](#) [30](#)
 - [6.6. Application Security](#) [31](#)
- [7. Challenges for a Secure IoT](#) [33](#)
 - [7.1. Constraints and Heterogeneous Communication](#) [33](#)
 - [7.1.1. Tight Resource Constraints](#) [33](#)
 - [7.1.2. Denial-of-Service Resistance](#) [34](#)
 - [7.1.3. End-to-End Security, protocol translation, and the role of middleboxes](#) [35](#)
 - [7.1.4. New network architectures and paradigm](#) [37](#)
 - [7.2. Bootstrapping of a Security Domain](#) [37](#)
 - [7.3. Operation](#) [37](#)
 - [7.3.1. End-to-End Security](#) [38](#)
 - [7.3.2. Group Membership and Security](#) [38](#)
 - [7.3.3. Mobility and IP Network Dynamics](#) [39](#)
 - [7.4. Software update](#) [39](#)

- [7.5. Verifying device behavior](#) [40](#)
- [7.6. End-of-life](#) [41](#)
- [7.7. Testing: bug hunting and vulnerabilities](#) [41](#)
- [7.8. Quantum-resistance](#) [42](#)
- [7.9. Privacy protection](#) [42](#)
- [7.10. Data leakage](#) [43](#)
- [7.11. Trustworthy IoT Operation](#) [44](#)
- [8. Conclusions and Next Steps](#) [44](#)
- [9. Security Considerations](#) [45](#)
- [10. IANA Considerations](#) [45](#)
- [11. Acknowledgments](#) [45](#)
- [12. Informative References](#) [45](#)
- Authors' Addresses [56](#)

1. Conventions and Terminology Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [[RFC2119](#)].

2. Introduction

The Internet of Things (IoT) denotes the interconnection of highly heterogeneous networked entities and networks 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). The term IoT was first coined by the Auto-ID center [[AUTO-ID](#)] in 1999. Since then, the development of the underlying concepts has ever increased its pace. Nowadays, the IoT presents a strong focus of research with various initiatives working on the (re)design, application, and usage of standard Internet technology in the IoT.

The introduction of IPv6 and web services as fundamental building blocks for IoT applications [[RFC6568](#)] promises to bring a number of basic advantages including: (i) a homogeneous protocol ecosystem that allows simple integration with Internet hosts; (ii) simplified development of very different appliances; (iii) an unified interface for applications, removing the need for application-level proxies. Such features greatly simplify the deployment of the envisioned scenarios ranging from building automation to production environments to personal area networks, in which very different things such as a temperature sensor, a luminaire, or an RFID tag might interact with each other, with a human carrying a smart phone, or with backend services.

This Internet Draft presents an overview of the security aspects of the envisioned all-IP architecture as well as of the lifecycle of an

IoT device, a "thing", within this architecture. In particular, we review the most pressing aspects and functionalities that are required for a secure all-IP solution.

With this, this Internet-Draft pursues several goals. First, we aim at presenting a comprehensive view of the interactions and relationships between an IoT application and security. Second, we aim at describing challenges for a secure IoT in the specific context of the lifecycle of a resource-constrained device. The final goal of this draft is to discuss the security considerations that need to be taken into consideration towards a secure IoT.

The rest of the Internet-Draft is organized as follows. [Section 3](#) depicts the lifecycle of a thing and gives general definitions for the main security building blocks within the IoT domain. In [Section 4](#), we discuss threats and methodologies for managing risks when designing a secure IoT system. [Section 5](#) reviews existing IP-based (security) protocols for the IoT and briefly summarizes existing guidelines and regulations in IoT security. [Section 6](#) proposes a number of illustrative security profiles describing how different applications would require distinct security solutions. [Section 7](#) identifies existing challenges for a secure IoT and discusses potential solutions. [Section 7](#) includes final remarks and conclusions.

[3. Motivation and background](#)

[3.1. The Thing Lifecycle](#)

The lifecycle of a thing refers to the operational phases of a thing in the context of a given application or use case. Figure 1 shows the generic phases of the lifecycle of a thing. This generic lifecycle is applicable to very different IoT applications and scenarios.

We consider an example, a Building Automation and Control (BAC) system, to illustrate the lifecycle and the meaning of these different phases. A BAC system consists of a network of interconnected nodes that performs various functions in the domains of HVAC (Heating, Ventilating, and Air Conditioning), lighting, safety etc. The nodes vary in functionality and a majority of them represent resource constrained devices such as sensors and luminaries. Some devices may also be battery operated or battery-less nodes, demanding for a focus on low energy consumption and on sleeping devices. In our example, the life of a thing starts when it is manufactured. Due to the different application areas (i.e., HVAC, lighting, safety) nodes are tailored to a specific task. It is therefore unlikely that one single manufacturer will create all nodes

in a building. Hence, interoperability as well as trust bootstrapping between nodes of different vendors is important. The thing is later installed and commissioned within a network by an installer during the bootstrapping phase. Specifically, the device identity and the secret keys used during normal operation are provided to the device during this phase. Different subcontractors may install different IoT devices for different purposes. Furthermore, the installation and bootstrapping procedures may not be a defined event but may stretch over an extended period of time. After being bootstrapped, the device and the system of things are in operational mode and execute the functions of the BAC system. During this operational phase, the device is under the control of the system owner. For devices with lifetimes spanning several years, occasional maintenance cycles may be required. During each maintenance phase, the software on the device can be upgraded or applications running on the device can be reconfigured. The maintenance tasks can thereby be performed either locally or from a backend system by means of an end-to-end connection. Depending on the operational changes of the device, it may be required to re-bootstrap at the end of a maintenance cycle. The device continues to loop through the operational phase and the eventual maintenance phase until the device is decommissioned at the end of its lifecycle. However, the end-of-life of a device does not necessarily mean that it is defective but rather denotes a need to replace and upgrade the network to next-generation devices in order to provide additional functionality. Therefore the device can be removed and re-commissioned to be used in a different system under a different owner by starting the lifecycle all over again.

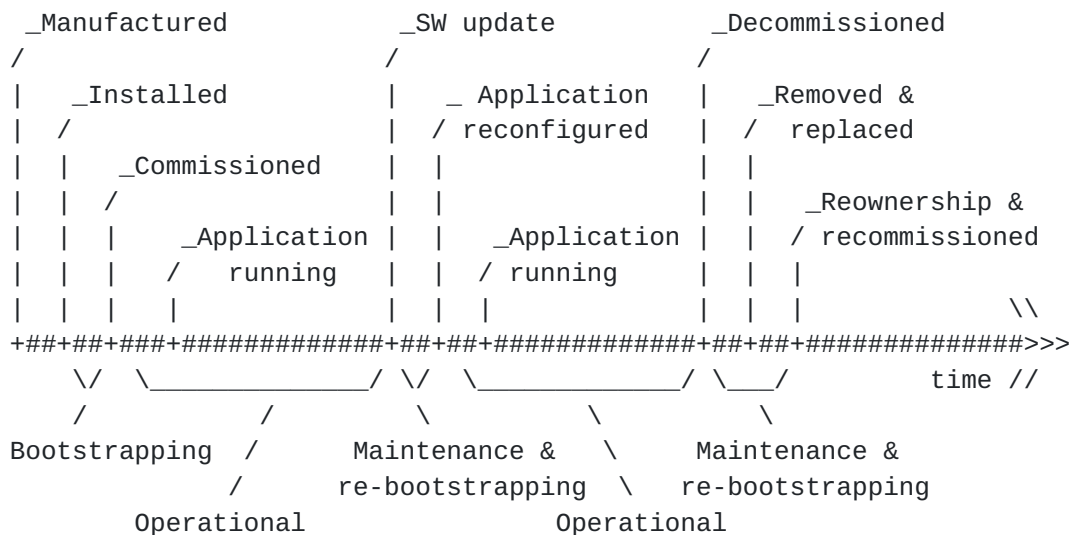


Figure 1: The lifecycle of a thing in the Internet of Things.

3.2. Security building blocks

Security is a key requirement in the IoT due to several reasons. First, an IoT systems enable very specific applications in which users are involved. A broken IoT system means that the privacy and safety of the users is endangered, this is key requirement in application areas such as critical infrastructure or health care. Second, a compromised IoT system means that an attacker altered the functionality of the devices of a given manufacturer, this not only affects the manufacturer's brand image in a negative way but can also leak information that is very valuable for the manufacturer, such as proprietary algorithms. Third, the impact of attacking the IoT goes beyond a specific device or isolated systems since compromised IoT systems can be misused at scale, e.g., performing a Distribute Denial of Service (DDoS) attack that limits the availability of the compromised system or even other IT networks. The fact that many IoT systems rely on standard IP protocols allows for easier system integration increasing the value of the realized use cases, but this also makes standard attacks applicable to a wide number of devices deployed in multiple systems. This results in new requirements regarding the implementation of security.

The term security subsumes a wide range of primitives, protocols, and procedures. In the first place, it includes the basic provision of security services including confidentiality, authentication, integrity, authorization, non-repudiation, and availability, and some augmented services, such as duplicate detection and detection of stale packets (timeliness). These security services can be implemented by a combination of cryptographic mechanisms, such as block ciphers, hash functions, or signature algorithms, and non-cryptographic mechanisms, which implement authorization and other security policy enforcement aspects. For each of the cryptographic mechanisms, a secure key management infrastructure is fundamental to handling the required cryptographic keys, whereas for security policy enforcement, one needs to properly codify authorizations as a function of device roles and a security policy engine that implements these authorization checks and that can implement changes hereto throughout the system's lifecycle.

In the particular context of the IoT, security must not only focus on the required security services, but also pay special attention to how these are realized in the overall system and how the security functionalities are executed. To this end, we consider five major "building blocks" to analyze and classify security aspects in the IoT:

1. The IoT security architecture: refers to the system elements involved in the management of the security relationships between

things and the way these security interactions are handled (e.g., centralized or distributed) during the lifecycle of a thing. For instance, a smart home could rely on a centralized key distribution center in charge of managing cryptographic keys, devices & users, access control and privacy policies.

2. The security model within a smart object: describes the way security parameters, key, processes, and applications are managed within a smart object. This includes aspects such as application process separation, secure storage of key materials, protection of algorithms, etc. For instance, some smart objects might have extremely limited resources and have limited capabilities to protect secret keys; in contrast, other devices used in critical applications, e.g., a pacemaker, would rely on methods to securely protect cryptographic keys and functionality making sure that an attacker having physical access to the device cannot modify its operation.
3. Security bootstrapping: denotes the process by which a thing securely joins an IoT system at a given location and point in time. For instance, bootstrapping of a connected camera can include the authentication and authorization of a device as well as the transfer of security parameters allowing for its trusted operation in a given network.
4. 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 an smart object, it also protects the network itself from malicious things. For instance, network security can include a number of mechanisms ranging from data link layer security, secure routing, and network layer security.
5. Application security: describes mechanisms to allow transfer of application data at the transport or upper layers (object security). For instance, assuming an smart object such as an environmental sensor connected to a backend system, it can mean the exchange of secure blocks of data such as measurements by the sensor or a software update. This data is exchanged end-to-end independently of communication pattern, for e.g through proxies or other store-and-forward mechanisms.

In our example, we consider a centralized architecture in which a configuration entity stores and manages the identities of the things associated with BAC system along with their cryptographic keys. During the bootstrapping phase, each thing executes the bootstrapping protocol with the configuration entity, thus obtaining the required device identities and some operational keying material. The security service on a thing in turn stores the received keying material for the network layer and application security mechanisms for secure communication. The criticality of the application requires an implementation of cryptographic algorithms that is resistant to side-channel attacks and the protection of the proprietary application-related algorithms executed in the device. Things can then securely communicate with each other during their operational phase by means of the employed network and application security mechanisms. Within the network, communication is protected by the network provider at MAC and network layer. At application layer, the communication between any smart object and the application server is protected end-to-end, ensuring the forward secrecy of the communication.

4. Managing Threats and Risks

This section explores security threats and vulnerabilities in the IoT and discusses how to manage risks.

Security threats have been analyzed in related IP protocols including HTTPS [[RFC2818](#)], COAP [[RFC7252](#)] 6LoWPAN [[RFC4919](#)], ANCP [[RFC5713](#)], DNS security threats [[RFC3833](#)], SIP [[RFC3261](#)], IPv6 ND [[RFC3756](#)], and PANA [[RFC4016](#)]. Nonetheless, the challenge is about their impacts on scenarios of the IoTs. In this section, we specifically discuss the threats that could compromise an individual thing, or network as a whole. Note that these set of threats might go beyond the scope of Internet protocols but we gather them here for the sake of completeness. We also note that these threats can be classified according to either (i) the thing's lifecycle phases (when does the threat occur?) or (ii) the security building blocks (which functionality is affected by the threat?). All these threats are summarized in Table 2.

1. Cloning of things: During the manufacturing process of a thing, an untrusted factory can easily clone the physical characteristics, firmware/software, or security configuration of the thing. Deployed things might also be compromised and their software reserve engineered allowing for cloning or software modifications. Such a cloned thing may be sold at a cheaper price in the market, and yet be able to function normally, as a genuine thing. For example, two cloned devices can still be associated and work with each other. In the worst case scenario, a cloned device can be used to control a genuine device or

perform an attack. One should note here, that an untrusted factory may also change functionality of the cloned thing, resulting in degraded functionality with respect to the genuine thing (thereby, inflicting potential damage to the reputation of the original thing manufacturer). Moreover, additional functionality can be implemented within the cloned thing, such as a backdoor.

2. Malicious substitution of things: During the installation of a thing, a genuine thing may be substituted with a similar variant of lower quality without being detected. The main motivation may be cost savings, where the installation of lower-quality things (e.g., non-certified products) may significantly reduce the installation and operational costs. The installers can subsequently resell the genuine things in order to gain further financial benefits. Another motivation may be to inflict damage to the reputation of a competitor's offerings.
3. Eavesdropping attack: During the commissioning of a thing into a network, it may be susceptible to eavesdropping, especially if operational keying materials, security parameters, or configuration settings, are exchanged in clear using a wireless medium or if used cryptographic algorithms are not suitable for the envisioned lifetime of the device and the system. After obtaining the keying material, the attacker might be able to recover the secret keys established between the communicating entities (e.g., H2T, T2Ts, or Thing to the backend management system), thereby compromising the authenticity and confidentiality of the communication channel, as well as the authenticity of commands and other traffic exchanged over this communication channel. When the network is in operation, T2T communication may be eavesdropped upon if the communication channel is not sufficiently protected or in the event of session key compromise due to a long period of usage without key renewal or updates.
4. Man-in-the-middle attack: Both the commissioning phase and operational phases may also be vulnerable to man-in-the-middle attacks, e.g., when keying material between communicating entities are exchanged in the clear and the security of the key establishment protocol depends on the tacit assumption that no third party is able to eavesdrop during the execution of this protocol. Additionally, device authentication or device authorization may be non-trivial, or may need support of a human decision process, since things usually do not have a-priori knowledge about each other and cannot always be able to differentiate friends and foes via completely automated mechanisms. Thus, even if the key establishment protocol

provides cryptographic device authentication, this knowledge on device identities may still need complementing with a human-assisted authorization step (thereby, presenting a weak link and offering the potential of man-in-the-middle attacks this way).

5. Firmware Replacement attack: When a thing is in operation or maintenance phase, its firmware or software may be updated to allow for new functionality or new features. An attacker may be able to exploit such a firmware upgrade by replacing the thing's software with malicious software, thereby influencing the operational behavior of the thing. For example, an attacker could add a piece of malicious code to the firmware that will cause it to periodically report the energy usage of the lamp to a data repository for analysis. Similarly, devices whose software has not been properly maintained and updated might contain vulnerabilities that might be exploited by attackers.
6. Extraction of private information: in the ambient environment the things (such as sensors, actuators, etc.) are usually physically unprotected and could easily be captured by an attacker. Such an attacker may then attempt to extract private information such as keys (e.g., device's key, private-key, group key), sensed data (e.g., healthcare status of a user), configuration parameters (e.g., the WiFi key), or proprietary algorithms (e.g., algorithm performing some data analytic task) from this thing. Compromise of a thing's unique key compromises communication channels of this particular thing and also compromise all data communicated over this channel.
7. Routing attack: As highlighted in [[ID-Daniel](#)], routing information in IoT can be spoofed, altered, or replayed, in order to create routing loops, attract/repel network traffic, extend/shorten source routes, etc. Other relevant routing attacks include 1) Sinkhole attack (or blackhole attack), where an attacker declares himself to have a high-quality route/path to the base station, thus allowing him to do manipulate all packets passing through it. 2) Selective forwarding, where an attacker may selectively forward packets or simply drop a packet. 3) Wormhole attack, where an attacker may record packets at one location in the network and tunnel them to another location, thereby influencing perceived network behavior and potentially distorting statistics, thus greatly impacting the functionality of routing. 4) Sybil attack, whereby an attacker presents multiple identities to other things in the network.
8. Privacy threat: The tracking of a thing's location and usage may pose a privacy risk to its users. An attacker can infer information based on the information gathered about individual

things, thus deducing behavioral patterns of the user of interest to him. Such information can subsequently be sold to interested parties for marketing purposes and targeted advertising.

9. Denial-of-Service attack: Typically, things have tight memory and limited computation, they are thus vulnerable to resource exhaustion attack. Attackers can continuously send requests to be processed by specific things so as to deplete their resources. This is especially dangerous in the IoTs since an attacker might be located in the backend and target resource-constrained devices in an Low-Latency Network (LLN). Additionally, DoS attack can be launched by physically jamming the communication channel, thus breaking down the T2T communication channel. Network availability can also be disrupted by flooding the network with a large number of packets. On the other hand, things compromised by attackers can be used to disrupt the operation of other networks or systems by means of a Distributed DoS attack.

The following table summarizes the above generic security threats and the potential point of vulnerabilities at different layers of the communication stack. We also include related RFCs and ongoing standardization efforts that include a threat model that might apply to the IoTs.

	Manufacturing	Installation/ Commissioning	Operation
Thing's Model	Device Cloning	Substitution ACE-OAuth(draft)	Privacy threat Extraction of private inform.
Application Layer		RFC2818 , RFC7252 OSCOAP(draft)	RFC2818 , Firmware replacement
Transport Layer		Eavesdropping & Man-in-the-middle attack, RFC7925	Eavesdropping Man-in-the-middle
Network Layer		RFC4919 , RFC5713 RFC3833 , RFC3756	RFC4919 , DoS attack Routing attack RFC3833
Physical Layer			DoS attack

Figure 3: Classification of threats according to the lifecycle phases and security building blocks.

Dealing with above threats and finding suitable security mitigations is challenging: there are very sophisticated threats that a very powerful attacker could use; also, new threats and exploits appear in a daily basis. Therefore, the existence of proper secure product creation processes that allow managing and minimizing risks during the lifecycle of the IoT devices is at least as important as being aware of the threats. A non-exhaustive list of relevant processes include:

1. A Business Impact Analysis (BIA) assesses the consequences of loss of basic security attributes, namely, confidentiality, integrity and availability in an IoT system. These consequences might include impact on data lost, sales lost, increased expenses, regulatory fines, customer dissatisfaction, etc. Performing a business impact analysis allow determining the business relevance of having a proper security design placing security in the focus.
2. A Risk Assessment (RA) analyzes security threats to the IoT system, considering their likelihood and impact, and deriving for each of them a risk level. Risks classified as moderate or high must be mitigated, i.e., security architecture should be able to deal with that threat bringing the risk to a low level. Note that threats are usually classified according to their goal: confidentiality, integrity, and availability. For instance, a specific threat to recover a symmetric-key used in the system relates to confidentiality.
3. A privacy impact assessment (PIA) aims at assessing Personal Identifiable Information (PII) that is collected, processed, or used in the IoT system. By doing so, the goals is to fulfill applicable legal requirements, determine risks and effects of the manipulation of PII, and evaluate proposed protections.
4. Procedures for incident reporting and mitigation refer to the methodologies that allow becoming aware of any security issues that affect an IoT system. Furthermore, this includes steps towards the actual deployment of patches that mitigate the identified vulnerabilities.

BIA, RA, and PIA are usually to be realized during the creation of a new IoT system, introduction of new technologies in the IoT system, or deployment of significant system upgrades. In general, it is recommended to re-assess them on a regular basis taking into account new use cases or threats.

5. State of the Art

This section is organized as follows. Section [Section 5.1](#) summarizes state of the art on IP-based systems, within IETF and in other standardization bodies. Section [Section 5.2](#) summarizes state of the art on IP-based security protocols and their usage. Section [Section 5.3](#) discusses guidelines for securing the IoT as proposed by other bodies. Section [Section 5.4](#) analyzes status of other relevant standards, in particular, those by NIST regarding IoT and IoT security.

5.1. IP-based IoT Protocols and Standards

Nowadays, there exists a multitude of control protocols for the IoT. For BAC systems, the ZigBee standard [[ZB](#)], BACNet [[BACNET](#)], or DALI [[DALI](#)] play key roles. Recent trends, however, focus on an all-IP approach for system control.

In this setting, a number of IETF working groups are designing new protocols for resource constrained networks of smart things. The 6LoWPAN working group [[WG-6LoWPAN](#)] concentrates on the definition of methods and protocols for the efficient transmission and adaptation of IPv6 packets over IEEE 802.15.4 networks [[RFC4944](#)].

The CoRE working group [[WG-CoRE](#)] provides a framework for resource-oriented applications intended to run on constrained IP network (6LoWPAN). One of its main tasks is the definition of a lightweight version of the HTTP protocol, the Constrained Application Protocol (CoAP) [[RFC7252](#)], that runs over UDP and enables efficient application-level communication for things.

In many smart object networks, the smart objects are dispersed and have intermittent reachability either because of network outages or because they sleep during their operational phase to save energy. In such scenarios, direct discovery of resources hosted on the constrained server might not be possible. To overcome this barrier, the CoRE working group is specifying the concept of a Resource Directory (RD) [[ID-rd](#)]. The Resource Directory hosts descriptions of resources which are located on other nodes. These resource descriptions are specified as CoRE link format [[RFC6690](#)] URIs.

While CoAP defines a standard communication protocol, a format for representing sensor measurements and parameters over CoAP is required. The Sensor Measurement Lists (SenML) [[ID-senml](#)] is a specification that is currently being written to define media types for simple sensor measurements and parameters. It has a minimalistic design so that constrained devices with limited computational

capabilities can easily encode their measurements and, at the same time, servers can efficiently collect large number of measurements.

In many IoT deployments, the resource-constrained smart objects are connected to the Internet via a gateway that is directly reachable. For example, an IEEE 802.11 Access Point (AP) typically connects the client devices to the Internet over just one wireless hop. However, some deployments of smart object networks require routing between the smart objects themselves. The IETF has therefore defined the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [[RFC6550](#)]. RPL provides support for multipoint-to-point traffic from resource-constrained smart objects towards a more resourceful central control point, as well as point-to-multipoint traffic in the reverse direction. It also supports point-to-point traffic between the resource-constrained devices. A set of routing metrics and constraints for path calculation in RPL are also specified [[RFC6551](#)].

In addition to defining a routing protocol, the IETF has also specified how IPv6 packets can be transmitted over various link layer protocols that are commonly employed for resource-constrained smart object networks. There is also ongoing work to specify IPv6 connectivity for a Non-Broadcast Multi-Access (NBMA) mesh network that is formed by IEEE 802.15.4 TimeSlotted Channel Hopping (TSCH) links [[ID-6tisch](#)]. Other link layer protocols for which IETF has specified or is currently specifying IPv6 support include Bluetooth [[RFC7668](#)], Digital Enhanced Cordless Telecommunications (DECT) Ultra Low Energy (ULE) air interface [[ID-6lodect](#)], and Near Field Communication (NFC) [[ID-6lonfc](#)].

JavaScript Object Notation (JSON) is a lightweight text representation format for structured data [[RFC7158](#)]. It is often used for transmitting serialized structured data over the network. IETF has defined specifications for encoding public keys, signed content, and claims to be transferred between two parties as JSON objects. They are referred to as JSON Web Keys (JWK) [[RFC7517](#)], JSON Web Signatures (JWS) [[RFC7515](#)] and JSON Web Token (JWT) [[RFC7519](#)].

An alternative to JSON, Concise Binary Object Representation (CBOR) [[RFC7049](#)] is a concise binary data format that is used for serialization of structured data. It is designed for extremely resource-constrained nodes and therefore it aims to provide a fairly small message size with minimal implementation code, and extensibility without the need for version negotiation. There is ongoing work to specify CBOR Object Signing and Encryption (COSE) [[ID-cose](#)], which would provide services similar to JWS and JWT.

The Light-Weight Implementation Guidance (LWIG) working group [[WG-LWIG](#)] is collecting experiences from implementers of IP stacks in

constrained devices. The working group has already produced documents such as [RFC7815](#) [[RFC7815](#)] which defines how a minimal Internet Key Exchange Version 2 (IKEv2) initiator can be implemented.

The Thing-2-Thing Research Group (T2TRG) [[RG-T2TRG](#)] is investigating the remaining research issues that need to be addressed in order to quickly turn the vision of IoT into a reality where resource-constrained nodes can communicate with each other and with other more capable nodes on the Internet.

Additionally industry alliances and other standardization bodies are creating constrained IP protocol stacks based on the IETF work. Some important examples of this include:

1. Thread [[Thread](#)]: Specifies the Thread protocol that is intended for a variety of IoT devices. It is an IPv6-based network protocol that runs over IEEE 802.15.4.
2. Industrial Internet Consortium [[IIoT](#)]: The consortium defines reference architectures and security frameworks for development, adoption and widespread use of Industrial Internet technologies based on existing IETF standards.
3. Internet Protocol for Smart Objects IPSO [[IPSO](#)]: The alliance specifies a common object model that would enable application software any device to interoperate with other conforming devices.
4. OneM2M [[OneM2M](#)]: The standards body defines technical and API specifications for IoT devices. It aims to create a service layer that can run on any IoT device hardware and software.
5. Open Connectivity Foundation (OCF) [[OCF](#)]: The foundation develops standards and certifications primarily for IoT devices that use Constrained Application Protocol (CoAP) as the application layer protocol.
6. Fairhair Alliance [[Fairhair](#)]: Specifies a middle-ware for IoT based Building Automation and Lighting System that can interoperate with different application standards for the professional domain.

[5.2.](#) Existing IP-based Security Protocols and Solutions

In the context of the IP-based IoT solutions, consideration of TCP/IP security protocols is important. There are a wide range of specialized as well as general-purpose key exchange and security solutions for the Internet domain such as IKEv2/IPsec [[RFC7296](#)], TLS

[[RFC5246](#)], DTLS [[RFC6347](#)], HIP [[RFC7401](#)], PANA [[RFC5191](#)], and EAP [[RFC3748](#)].

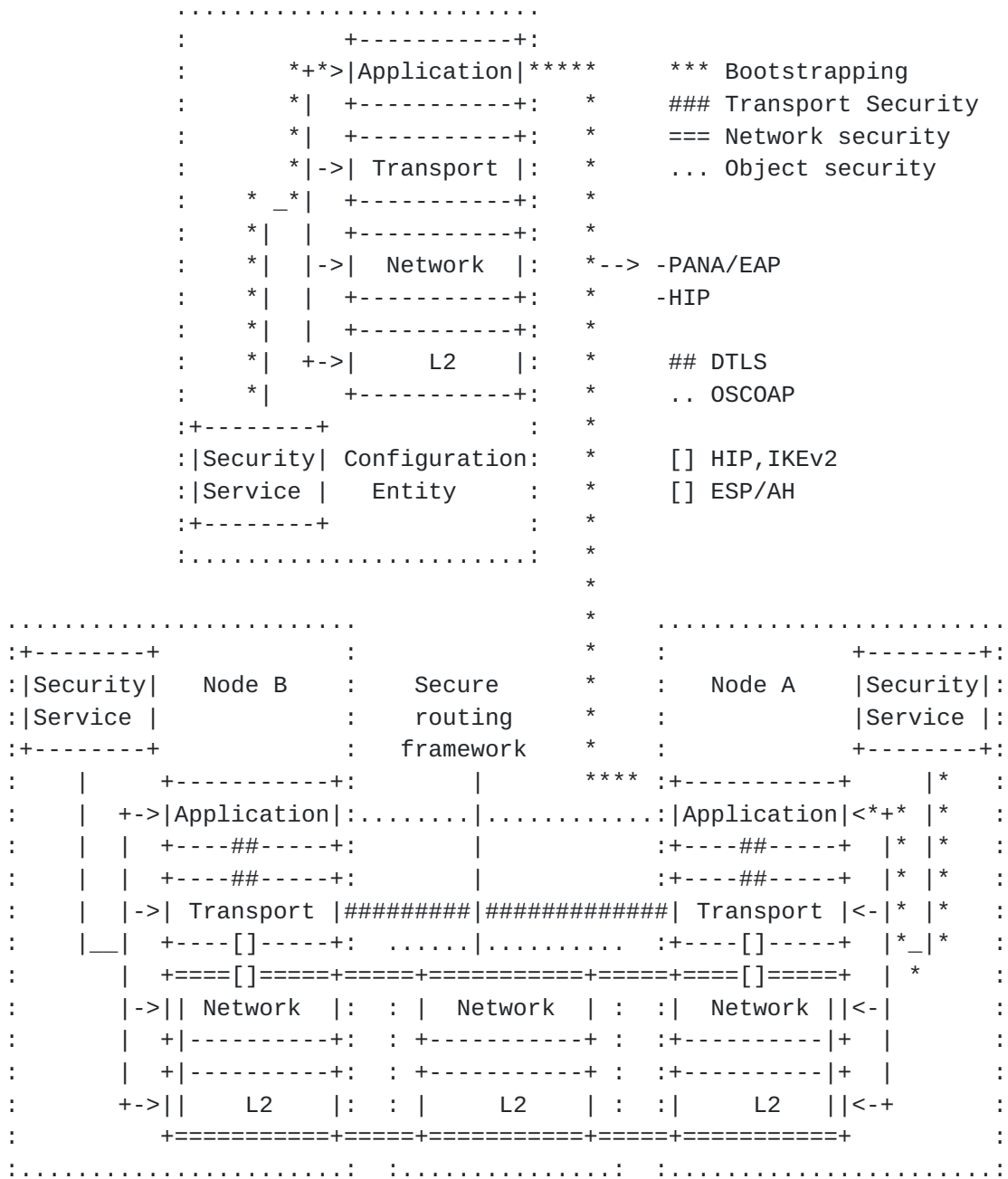
There is ongoing work to define an authorization and access-control framework for resource-constrained nodes. The Authentication and Authorization for Constrained Environments (ACE) [[WG-ACE](#)] working group is defining a solution to allow only authorized access to resources that are hosted on a smart object server and are identified by a URI. The current proposal [[ID-aceoauth](#)] is based on the OAuth 2.0 framework [[RFC6749](#)].

The CoAP base specification [[RFC7252](#)] provides a description of how DTLS can be used for securing CoAP. It proposes three different modes for using DTLS: the PreSharedKey mode, where nodes have pre-provisioned keys for initiating a DTLS session with another node, RawPublicKey mode, where nodes have asymmetric-key pairs but no certificates to verify the ownership, and Certificate mode, where public keys are certified by a certification authority. An IoT implementation profile [[RFC7925](#)] is defined for TLS version 1.2 and DTLS version 1.2 that offers communications security for resource-constrained nodes.

There is also work on Object Security based CoAP protection mechanism being defined in OSCOAP [[ID-OSCOAP](#)].

Migault et al. [[ID-dietesp](#)] are working on a compressed version of IPsec so that it can easily be used by resource-constrained IoT devices. They rely on the Internet Key Exchange Protocol version 2 (IKEv2) for negotiating the compression format.

Figure 3 depicts the relationships between the discussed protocols in the context of the security terminology introduced in [Section 3](#).



Relationships between IP-based security protocols.

Figure 4

The Internet Key Exchange (IKEv2)/IPsec and the Host Identity protocol (HIP) reside at or above the network layer in the OSI model. Both protocols are able to perform an authenticated key exchange and set up the IPsec transforms for secure payload delivery. Currently, there are also ongoing efforts to create a HIP variant coined Diet HIP [ID-HIP] that takes lossy low-power networks into account at the authentication and key exchange level.

Transport Layer Security (TLS) and its datagram-oriented variant DTLS secure transport-layer connections. TLS provides security for TCP and requires a reliable transport, while DTLS secures and uses datagram-oriented protocols such as UDP. Both protocols are intentionally kept similar and share the same ideology and cipher suites.

The Extensible Authentication Protocol (EAP) is an authentication framework supporting multiple authentication methods. EAP runs directly over the data link layer and, thus, does not require the deployment of IP. It supports duplicate detection and retransmission, but does not allow for packet fragmentation. The Protocol for Carrying Authentication for Network Access (PANA) is a network-layer transport for EAP that enables network access authentication between clients and the network infrastructure. In EAP terms, PANA is a UDP-based EAP lower layer that runs between the EAP peer and the EAP authenticator.

In addition, there is also new activities in IETF and W3C to define security protocols better tailored to IoT or for specific deployment situations.

5.3. IoT Security Guidelines

Recent large scale Denial of Service (DoS) Attacks on the Internet Infrastructure from compromised IoT devices has prompted many different standards bodies and consortia to provide guidelines for developers and the Internet community at large to build secure IoT devices and services. A subset of the different guidelines and ongoing projects are as follows:

1. GSMA IoT security guidelines [[GSMAsecurity](#)]: GSMA has published a set of security guidelines for the benefit of new IoT product and service providers. The guideline are aimed at device manufacturers, service providers, developers and network operators. An enterprise can complete an IoT Security Self-Assessment to demonstrate that its products and services are aligned with the security guidelines of the GSMA.
2. BITAG Internet of Things (IoT) Security and Privacy Recommendations [[BITAG](#)]: Broadband Internet Technical Advisory Group (BITAG) has also published recommendations for ensuring security and privacy of IoT device users. BITAG observes that many IoT devices are shipped from the factory with software that is already outdated and vulnerable. The report also states that many devices with vulnerabilities will not be fixed either because the manufacturer does not provide updates or because the user does not apply them. The recommendations include that IoT

devices should function without cloud and Internet connectivity, and that all IoT devices should have methods for automatic secure software updates.

3. CSA New Security Guidance for Early Adopters of the IoT [[CSA](#)]: The Cloud Security Alliance (CSA) recommendations for early adopters of IoT encourages enterprises to implement security at different layers of the protocol stack. It also recommends implementation of an authentication/authorization framework for IoT deployments. A complete list of recommendations is available in the report [[CSA](#)].
4. U.S. Department of Homeland Security [[DHS](#)]: DHS has put forth six strategic principles that would enable IoT developers, manufacturers, service providers and consumers to maintain security as they develop, manufacture, implement or use network-connected IoT devices.
5. NIST [[NIST-Guide](#)]: The NIST special publication urges enterprise and US federal agencies to address security throughout the systems engineering process. The publication builds upon the ISO/IEC/IEEE 15288 standard and augments each process in the system lifecycle with security enhancements.
6. NIST [[nist_lightweight_project](#)]: NIST is running a project on lightweight cryptography with the purpose of: (i) identifying application areas for which standard cryptographic algorithms are too heavy, classifying them according to some application profiles to be determined; (ii) determining limitations in those existing cryptographic standards; and (iii) standardizing lightweight algorithms that can be used in specific application profiles.
7. OWASP [[OWASP](#)]: Open Web Application Security Project (OWASP) provides security guidance for IoT manufactures, developers and consumers. OWASP also includes guidelines for those who intend to test and analyze IoT devices and applications.
8. IoT Security foundation [[IoTSecFoundation](#)]: IoT security foundation has published a document that enlists various considerations that need to be taken into account when developing IoT applications. For example, the document states that IoT device could use hardware-root of trust to ensure that only authorized software runs on the device.
9. NHTSA [[NHTSA](#)]: The US National Highway Traffic Safety Administration provides a set of non-binding guidance to the automotive industry for improving the cyber security of

vehicles. While some of the guidelines are general, the document provides specific recommendations for the automotive industry such as how various automotive manufacturer can share cyber security vulnerabilities discovered.

10. BCP for IoT devices [[ID-Moore](#)]: This Internet draft provides a list of minimum requirements that vendors of Internet of Things (IoT) devices should take into account while developing applications, services and firmware updates in order to reduce the frequency and severity of security incidents that arise from compromised IoT devices.
11. ENISA [[ENISA ICS](#)]: The European Union Agency for Network and Information Security published a document on communication network dependencies for ICS/SCADA systems in which security vulnerabilities, guidelines and general recommendations are summarized.

Other guideline and recommendation documents may exist or may later be published. This list should be considered non-exhaustive.

[5.4.](#) Guidelines and IoT Security Regulations

Despite the need for security in the Internet is nothing new and multiple guidelines exist, the fact is that many IoT devices and systems are not fully secure. There are multiple reasons for this. For instance, some manufactures focus on delivering a product without paying enough attention to the delivered security level, lack of expertise or budget. This, however, poses a severe threat when such devices are deployed. The vast amount of devices and their inherent mobile nature also implies that an initially secure system can become insecure if a compromised device gains access to the system at some point of time. Even if all devices in a given environment are secure, it does not prevent external (passive) attacks originating due to insecure devices.

Recently [[FCC](#)] the FCC has stated the need for higher regulation for IoT systems. In fact this might be a missing component, at least in Federal Information Systems (FIS). Today, security in US FIS is regulated according to Federal Information Security Management Act (FISMA). From this law, NIST derived a number of documents to establish how to categorize FIS and determine minimum security requirements (FIPS-PUB-199 and FIPS-PUB-200). Minimum security requirements for FIS are specified in NIST SP 800-53r4. However, it is very likely that existing regulations do not take into account the specific challenges of IoT devices and networks.

Even if such a regulation is put in place, the question is how such a regulation can be applied in practice to non-federal deployments, such as industrial, homes, offices, or smart cities. Each of them exhibits unique features, involves very diverse types of users, has different operational requirements, and combines IoT devices from multiple manufacturers. Therefore future regulations should consider such diverse deployment scenarios.

6. IoT Security Profiles

There is a wide range of IoT applications including building automation systems, healthcare, smart cities, logistics, etc. For each of these applications, properties such as device capability, network infrastructure, or available security services can be completely different. Furthermore, each of those applications is featured by a different number of actors deployed in very different environments and with very different purposes.

Consequently, when a Business Impact Analysis or Risk Assessment is performed, not only the types of threats will be different, but also their likelihood and potential impact. This determines that different applications tend to require different or complementary types of security mechanisms mitigating the identified risks.

For example, IoT applications may have different needs regarding authentication and confidentiality. While some application might not require any confidentiality at all, others might require strong end-to-end confidentiality. In terms of secure bootstrapping of keys, some applications might assume the existence and online availability of a central key-distribution-center (KDC) within the network to distribute and manage keys; while other applications cannot rely on such a central party or on their availability.

This section describes some exemplary security profiles fitting the security needs of applications with the same characteristics and requirements. This approach is similar to that in the security profiles in [[nist lightweight project](#)]. Such security profiles can help to (i) guide the design process of different application types by identifying open gaps; (ii) allow for later interoperability; and (iii) prevent possible security misconfiguration. Each security profile is identified by:

1. a short description,
2. an exemplary application that might use/require such a security profile,

3. the security requirements for each of the above security aspects according to our classification in [Section 3.2](#).

These security profiles can serve to guide the standardization process, since these explicitly describe the basic functionalities and protocols required to get different use cases up and running. They can allow for later interoperability since different manufacturers can describe the implemented security profile in their products. Finally, the security profiles can avoid possible security misconfiguration, since each security profile can be bound to a different application domain so that security protocols are clearly defined and under which circumstances they are applied.

We compare the security capabilities in each of the security profiles according to security building blocks introduced [Section 3.2](#), namely:

1. Security architecture,
2. Security model,
3. Security bootstrapping,
4. Network security, and
5. Application security.

IMPORTANT: Note that each of these exemplary profiles aims at summarizing the required security requirements for different exemplary application areas and at providing a set of initial security features. In other words, these profiles reflect the need for different security configurations, depending on the threat and trust models of the underlying applications. In this sense, this section does not provide an overview of existing protocols as done in previous sections, but it rather explicitly describes what should be in place to ensure secure system operation. Observe also that this list of security profiles is not exhaustive and that it should be considered just as an example not related to existing legal regulations for any existing application.

The remainder of this section is organized as follows.

Section [Section 6.1](#) first describes four generic security profiles and discuss how different applications of IP networks, e.g., 6LoWPAN/CoAP networks, involve different security needs. The following five subsections summarize the expected security features or capabilities for each the security profile with regards to "Security Architecture", "Security Model", "Security Bootstrapping", "Network Security", and "Application Security".

6.1. Profiles Security Profiles

We consider four generic security profiles as summarized in the table below:

	Exemplary Application	Description
SecProf_1	Home usage	Enables operation between home things without interaction with central device
SecProf_2	Managed Home usage	Enables operation between home things. Interaction with a central and local device is possible
SecProf_3	Industrial usage	Enables operation between things. Relies on central (local or backend) device for security
SecProf_4	Advanced Industrial usage	Enables ad-hoc operation between things and relies on central device or on a collection of control devices

Figure 5: Security profiles and application areas.

The classification in the table considers different potential applications in which security mechanism are chosen according to the operational features (network size, existence of a central device, connectivity to the Internet, importance of the exchanged information, etc.) and threat model (what are the assets that an attacker looks for). As already pointed out, this set of scenarios is just exemplary and they should be further discussed based on a broader consensus.

The security suite (SecProf_1) is catered for environments in which IP protocols (e.g., 6LoWPAN/CoAP) can be used to enable communication between things in an ad-hoc manner and the security requirements are minimal. An example, is a home application in which two devices should exchange information and no further connection with other devices (local or with a backend) is required. In this scenario, value of the exchanged information is low and usually happens in a confined room, thus, it is possible to have a short period of time during which initial secrets can be exchanged in the clear. Due to this fact, there is no requirement to enable devices from different manufacturers to inter operate in a secure way (keys are just exchanged). The expected network size of applications using this

profile is expected to be small such that the provision of network security, e.g., secure routing, is of low importance.

The next security suite (SecProf_2) represents an evolution of SecProf_1 in which, e.g., home devices, can be managed. A first possibility for the securing domain management refers to the creation of a centrally managed security domain without any connectivity to the Internet. The central device used for management can serve as, e.g., a key distribution center including policies for key update, storage, etc. The presence of a central device can help in the management of larger networks. Network security becomes more relevant in this scenario since the IP network (e.g., 6LoWPAN/CoAP network) can be prone to Denial of Service attacks (e.g., flooding if L2 is not protected) or routing attacks. Similarly, the network of devices could also be the source of a DDoS attack and a central device connecting to the Internet can block traffic of ongoing attacks.

SecProf_3 considers that a central device is always required for managing the system. Example applications of this profile include building control and automation, sensor networks for industrial use, environmental monitoring, etc. As before, the manager can be located in the same network (e.g., 6LoWPAN/CoAP network) and handle key management. In this case, the first association of devices to the network is required to be done in a secure way, i.e., requiring authentication and authorization. This step can involve the secure transmission of keying materials used for network security at different layers. The information exchanged in the network is considered to be valuable and it should be protected in the sense of pairwise links. Commands should be secured and broadcast should be secured with entity authentication [[RFC7390](#)]. Network should be protected from routing attacks. A further extension to this use case is to allow for remote management. A "backend manager" is in charge of securely managing SW or information exchanged or collected within the network, e.g., a 6LoWPAN/CoAP network. This requires connection of devices to the Internet over a 6LBR involving a number of new threats that were not present before. A list of potential attacks include: resource-exhaustion attacks from the Internet; amplification attacks; trust issues related a HTTP-CoAP proxy [[ID-proHTTCoAP](#)], etc. This use case requires protecting the communication from a device in the backend to a device in the IP network, e.g., a 6LoWPAN/CoAP network, end-to-end. This use case also requires measures to provide the 6LBR with the capability of dropping fake requests coming from the Internet. This becomes especially challenging when the 6LBR is not trusted and access to the exchanged information is limited; and even more in the case of a HTTP-CoAP proxy since protocol translation is required. This use case should take care of protecting information accessed from the backend due to privacy

issues (e.g., information such as type of devices, location, usage, type and amount of exchanged information, or mobility patterns can be gathered at the backend threatening the privacy sphere of users) so that only required information is disclosed.

The last security suite (SecProf_4) essentially represents interoperability of all the security profiles defined previously. It considers applications with some additional requirements regarding operation such as: (i) ad-hoc establishment of security relationships between things (potentially from different manufacturers) in non-secure environments or (ii) dynamic roaming of things between different IP network security domains. Such operational requirements pose additional security requirements, e.g., in addition to secure bootstrapping of a device within an IP, e.g., 6LowPan/CoAP, security domain and the secure transfer of network operational key, there is a need to enable inter-domains secure communication to facilitate data sharing. In this scenario, there is also a higher pressure to ensure that an attacker cannot compromise deployed devices and extract or modify any type of private data such as cryptographic keys, data, or proprietary algorithms.

6.2. Security Architecture

Most things might be required to support both centralized and distributed operation patterns. Distributed thing-to-thing communication might happen on demand, for instance, when two things form an ad-hoc security domain to cooperatively fulfill a certain task. Likewise, nodes may communicate with a backend service located in the Internet without a central security manager. The same nodes may also be part of a centralized architecture with a dedicated node being responsible for the security management for group communication between things in the IoT domain. In today's IoT, most common architectures are fully centralized in the sense that all the security relationships within a segment are handled by a central party. In the ZigBee standard, this entity is the trust center. Current proposals for 6LOWPAN/CoRE identify the 6LOWPAN Border Router (6LBR) as such a device.

A centralized architecture allows for central management of devices and keying materials as well as for the backup of cryptographic keys. However, it also imposes some limitations. First, it represents a single point of failure. This is a major drawback, e.g., when key agreement between two devices requires online connectivity to the central node. Second, it limits the possibility to create ad-hoc security domains without dedicated security infrastructure. Third, it codifies a more static world view, where device roles are cast in stone, rather than a more dynamic world view that recognizes that

networks and devices, and their roles and ownership, may change over time (e.g., due to device replacement and hand-over of control).

Decentralized architectures, on the other hand, allow creating ad-hoc security domains that might not require a single online management entity and are operative in a much more stand-alone manner. The ad-hoc security domains can be added to a centralized architecture at a later point in time, allowing for central or remote management.

The choice of security architecture has many implications regarding key management, access control, or security scope. A distributed (or ad-hoc) architecture means that security relationships between things are setup on the fly between a number of objects and kept in a decentralized fashion, i.e., there is no central authority that can interfere with the system operation. A locally centralized security architecture means that a central device, e.g., the 6LBR, handles the keys for all the devices in the security domain. Alternatively, a central security architecture could also refer to the fact that smart objects are managed from the backend. It can also refer to a public-key infrastructure used to manage identities and digital certificates associated to the different devices.

The security architecture for the different security profiles is classified as follows.

	Description
SecProf_1	Distributed
SecProf_2	Distributed able to move centralized (local)
SecProf_3	Centralized (local &/or backend)
SecProf_4	Distributed & centralized (local &/or backend)

Figure 6: Security architectures in different security profiles.

In "SecProf_1", management mechanisms for the distributed assignment and management of keying materials is required. Since this is a very simple use case, access control to the formed security domain can be enabled by means of a common secret known to all devices. In the next security suite (SecProf_2), a central device can assume key management responsibilities and handle the access to the network. The last two security suites (SecProf_3 and SecProf_4) further allow for the management of devices or some operational keying materials from the backend. Identity and public-key management can be realized

by means of a public-key infrastructure or a more decentralized solution based on block-chain.

6.3. Security Model

While some applications might involve very resource-constrained things such as, e.g., a humidity, pollution sensor, other applications might target more powerful devices aimed at more exposed applications. Security parameters such as keying materials, certificates, etc. must be protected in the thing, for example by means of tamper-resistant hardware. Keys may be shared across a thing's networking stack to provide authenticity and confidentiality in each networking layer. This would minimize the number of key establishment/agreement handshake and incurs less overhead for constrained thing. While more advanced applications may require key separation at different networking layers, and possibly process separation and sandboxing to isolate one application from another. In this sense, this section reflects the fact that different applications require different sets of security mechanisms. A very important requirement refers to the protection of application related parameters, data, and algorithms running on a smart object that should be protected from extraction or manipulation.

	Description
SecProf_1	No tamper resistant Sharing keys between layers
SecProf_2	No tamper resistant Sharing keys between layers
SecProf_3	Tamper resistant Key and process separation Data and algorithm protection from manipulation/copy
SecProf_4	(no) Tamper resistant Sharing keys between layers/Key and process separation Sandbox Data and algorithm protection from manipulation/copy

Figure 7: Thing security models in different security profiles.

[6.4.](#) Security Bootstrapping and Management

Bootstrapping refers to the process by which a thing initiates its life within a security domain and includes the initialization of secure and/or authentic parameters bound to the thing and at least one other device in the network. Here, different mechanisms may be used to achieve confidentiality and/or authenticity of these parameters, depending on deployment scenario assumptions and the communication channel(s) used for passing these parameters. The simplest mechanism for initial set-up of secure and authentic parameters is via communication in the clear using a physical interface (USB, wire, chip contact, etc.). Here, one commonly assumes this communication channel is secure, since eavesdropping and/or manipulation of this interface would generally require access to the physical medium and, thereby, to one or both of the devices themselves. This mechanism was used with the so-called original "resurrecting duckling" model, as introduced in [[PROC-Stajano-99](#)]. This technique may also be used securely in wireless, rather than wired, set-ups, if the prospect of eavesdropping and/or manipulating this channel are dim (a so-called "location-limited" channel [[PROC-Smetters-04](#)][[PROC-Smetters-02](#)]). Examples hereof include the communication of secret keys in the clear using near field communication (NFC) - where the physical channel is purported to have very limited range (roughly 10cm), thereby thwarting eavesdropping by far-away adversarial devices, and in-the-clear communication during a small time window (triggered by, e.g., a button-push) - where eavesdropping is presumed absent during this small time window. With the use of public-key based techniques, assumptions on the communication channel can be relaxed even further, since then the cryptographic technique itself provides for confidentiality of the channel set-up and the location-limited channel - or use of certificates - rules out man-in-the-middle attacks, thereby providing authenticity [[PROC-Smetters-02](#)]. The same result can be obtained using password-based public-key protocols [[SPEKE](#)], where authenticity depends on the (weak) password not being guessed during execution of the protocol.

It should be noted that while most of these techniques realize a secure and authentic channel for passing parameters, these generally do not provide for explicit authorization. As an example, with use of certificate-based public-key based techniques, one may obtain hard evidence on whom one shares secret and/or authentic parameters with, but this does not answer the question as to whether one wishes to share this information at all with this specifically identified device (the latter usually involves a human-decision element). Thus, the bootstrapping mechanisms above should generally be complemented by mechanisms that regulate (security policies for) authorization. Furthermore, the type of bootstrapping is very related to the

required type of security architecture. Distributed bootstrapping means that a pair of devices can setup a security relationship on the fly, without interaction with a central device elsewhere within the system. In many cases, it is handy to have a distributed bootstrapping protocol based on existing security protocols (e.g., DTLS in CoAP) required for other purposes: this reduces the amount of required software. A centralized bootstrapping protocol is one in which a central device manages the security relationships within a network. This can happen locally, e.g., handled by the 6LBR, or remotely, e.g., from a server connected via the Internet. The security bootstrapping for the different security profiles is as follows.

+-----+ Description +-----+	
SecProf_1	* Distributed, (e.g., Resurrecting duckling) * First key distribution happens in the clear
SecProf_2	* Distributed, (e.g., Resurrecting duckling) * Centralized (local), 6LBR acts as KDC * First key distribution occurs in the clear, if the KDC is available, the KDC can manage network access
SecProf_3	* 6LBR acts as KDC. It handles node joining, provides them with keying material from L2 to application layers * Bootstrapping occurs in a secure way - either in secure environment or the security mechanisms ensure that eavesdropping is not possible. * KDC and backend can implement secure methods for network access
SecProf_4	* As in SecProf_3.

Figure 8: Security bootstrapping methods in different security profiles

6.5. Network Security

Network security refers to the mechanisms used to ensure the secure transport of network packets such as 6LoWPAN frames. This involves a multitude of issues ranging from secure discovery, frame authentication, routing security, detection of replay, secure group communication, etc. Network security is important to thwart potential attacks such as denial-of-service (e.g., through message flooding) or routing attacks.

The Internet Draft [[ID-Tsao](#)] presents a very good overview of attacks and security needs classified according to the confidentiality, integrity, and availability needs. A potential limitation is that there exist no differentiation in security between different use cases and the framework is limited to L3. The security suites gathered in the present ID aim at solving this by allowing for a more flexible selection of security needs at L2 and L3.

	Description
SecProf_1	* Network key creating a home security domain at L2 ensuring authentication and freshness of exchanged data * Secure multicast does not ensure origin authentication * No need for secure routing at L3
SecProf_2	* Network key creating a home security domain at L2 ensuring authentication and freshness of exchanged data * Secure multicast does not ensure origin authentication * No need for secure routing at L3
SecProf_3	* Network key creating an industry security domain at L2 ensuring authentication and freshness of exchanged data * Secure routing needed (integrity & availability) at L3 within 6LoWPAN/CoAP * Secure multicast requires origin authentication
SecProf_4	* Network key creating an industry security domain at L2 ensuring authentication and freshness of exchanged data * Inter-domain authentication/secure handoff * Secure routing needed at L3 * Secure multicast requires origin authentication * 6LBR (HTTP-CoAP proxy) requires verification of forwarded messages and messages leaving or entering the 6LoWPAN/CoAP network.

Figure 9: Network security needs in different security profiles

6.6. Application Security

In the context of 6LoWPAN/CoAP networks, application security refers firstly to the configuration of an application layer protocol, such as DTLS, to protect the exchanged information. It further refers to the measures required in potential translation points (e.g., a (HTTP-CoAP) proxy) where information can be collected and the privacy sphere of users in a given security domain is endangered.

Application security for the different security profiles is as follows.

	Description
SecProf_1	-
SecProf_2	* DTLS is used for end-to-end application security between management device and things and between things * DTLS ciphersuites configurable to provide confidentiality and/or authentication and/or freshness * Key transport and policies for generation of session keys are required
SecProf_3	* Requirements as in SecProf_2 and * DTLS is used for end-to-end application security between management device and things and between things * Communication between KDC and each thing secured by pairwise keys * Group keys for communication in a group distributed by KDC * Privacy protection should be provided in translation points
SecProf_4	* Requirements as in SecProf_3 and * TLS or DTLS can be used to send commands from the backend to the 6LBR or things in a 6LoWPAN/CoAP network * End-to-end secure connectivity from backend required * Secure broadcast in a network from backend required

Figure 10: Application security methods in different security profiles

The first two security profiles do not include any security at the application layer. The reason is that, in the first case, security is not provided and, in the second case, it seems reasonable to provide basic security at L2. In the third security profile (SecProf_2), DTLS becomes the way of protecting messages at application layer between things and with the KDC running on a 6LBR. A key option refers to the capability of easily configuring DTLS to provide a subset of security services (e.g., some applications do not require confidentiality) to reduce the impact of security in the system operation of resource-constrained things. In addition to basic key management mechanisms running within the KDC, communication protocols for key transport or key update are required. These protocols could be based on DTLS. The next security suite

(SecProf_3) requires pairwise keys for communication between things within the security domain. Finally, privacy protection should be taken into account to limit access to valuable information -- such as identifiers, type of collected data, traffic patterns -- in potential translation points (proxies) or in the backend. The last security suite (SecProf_4) further extends the previous set of requirements considering security mechanisms to deal with translations between TLS and DTLS or for the provision of secure multicast within a 6LoWPAN/CoAP network from the backend.

7. Challenges for a Secure IoT

In this section, we take a closer look at the various security challenges in the operational and technical features of the IoT and then discuss how existing Internet security protocols cope with these technical and conceptual challenges through the lifecycle of a thing. Figure 2 summarizes which requirements need to be met in the lifecycle phases as well as some of the considered protocols. This discussion should neither be understood as a comprehensive evaluation of all protocols, nor can it cover all possible aspects of IoT security. Yet, it aims at showing concrete limitations of existing Internet security protocols in some areas rather than giving an abstract discussion about general properties of the protocols. In this regard, the discussion handles issues that are most important from the authors' perspectives.

7.1. Constraints and Heterogeneous Communication

Coupling resource constrained networks and the powerful Internet is a challenge because the resulting heterogeneity of both networks complicates protocol design and system operation. In the following we briefly discuss the resource constraints of IoT devices and the consequences for the use of Internet Protocols in the IoT domain.

7.1.1. Tight Resource Constraints

The IoT is a resource-constrained network that relies on lossy and low-bandwidth channels for communication between small nodes, regarding CPU, memory, and energy budget. These characteristics directly impact the threats to and the design of security protocols for the IoT domain. First, the use of small packets, e.g., IEEE 802.15.4 supports 127-byte sized packets at the physical layer, may result in fragmentation of larger packets of security protocols. This may open new attack vectors for state exhaustion DoS attacks, which is especially tragic, e.g., if the fragmentation is caused by large key exchange messages of security protocols. Moreover, packet fragmentation commonly downgrades the overall system performance due to fragment losses and the need for retransmissions. For instance,

fate-sharing packet flight as implemented by DTLS might aggravate the resulting performance loss.

The size and number of messages should be minimized to reduce memory requirements and optimize bandwidth usage. In this context, layered approaches involving a number of protocols might lead to worse performance in resource-constrained devices since they combine the headers of the different protocols. In some settings, protocol negotiation can increase the number of exchanged messages. To improve performance during basic procedures such as, e.g., bootstrapping, it might be a good strategy to perform those procedures at a lower layer.

Small CPUs and scarce memory limit the usage of resource-expensive crypto primitives such as public-key cryptography as used in most Internet security standards. This is especially true, if the basic crypto blocks need to be frequently used or the underlying application demands a low delay.

Independently from the development in the IoT domain, all discussed security protocols show efforts to reduce the cryptographic cost of the required public-key-based key exchanges and signatures with ECC [[RFC5246](#)][[RFC5903](#)][[RFC7401](#)][ID-HIP]. Moreover, all protocols have been revised in the last years to enable crypto agility, making cryptographic primitives interchangeable. However, these improvements are only a first step in reducing the computation and communication overhead of Internet protocols. The question remains if other approaches can be applied to leverage key agreement in these heavily resource-constrained environments.

A further fundamental need refers to the limited energy budget available to IoT nodes. Careful protocol (re)design and usage is required to reduce not only the energy consumption during normal operation, but also under DoS attacks. Since the energy consumption of IoT devices differs from other device classes, judgements on the energy consumption of a particular protocol cannot be made without tailor-made IoT implementations.

7.1.2. Denial-of-Service Resistance

The tight memory and processing constraints of things naturally alleviate resource exhaustion attacks. Especially in unattended T2T communication, such attacks are difficult to notice before the service becomes unavailable (e.g., because of battery or memory exhaustion). As a DoS countermeasure, DTLS, IKEv2, HIP, and Diet HIP implement return routability checks based on a cookie mechanism to delay the establishment of state at the responding host until the address of the initiating host is verified. The effectiveness of

these defences strongly depends on the routing topology of the network. Return routability checks are particularly effective if hosts cannot receive packets addressed to other hosts and if IP addresses present meaningful information as is the case in today's Internet. However, they are less effective in broadcast media or when attackers can influence the routing and addressing of hosts (e.g., if hosts contribute to the routing infrastructure in ad-hoc networks and meshes).

In addition, HIP implements a puzzle mechanism that can force the initiator of a connection (and potential attacker) to solve cryptographic puzzles with variable difficulties. Puzzle-based defence mechanisms are less dependent on the network topology but perform poorly if CPU resources in the network are heterogeneous (e.g., if a powerful Internet host attacks a thing). Increasing the puzzle difficulty under attack conditions can easily lead to situations, where a powerful attacker can still solve the puzzle while weak IoT clients cannot and are excluded from communicating with the victim. Still, puzzle-based approaches are a viable option for sheltering IoT devices against unintended overload caused by misconfiguration or malfunctioning things.

7.1.3. End-to-End Security, protocol translation, and the role of middleboxes

The term end-to-end security offers multiple interpretations. Here, we consider end-to-end security in the context end-to-end IP connectivity. Note that this does not necessarily mean from sensor to actuator.

Even though 6LoWPAN and CoAP progress towards reducing the gap between Internet protocols and the IoT, they do not target protocol specifications that are identical to their Internet counterparts due to performance reasons. Hence, more or less subtle differences between IoT protocols and Internet protocols will remain. While these differences can easily be bridged with protocol translators at middleboxes, they become major obstacles if end-to-end security measures between IoT devices and Internet hosts are used.

Regarding end-to-end security in the context of confidentiality and integrity protection, the packets are processed applying message authentication codes or encryption. These protection methods render the protected parts of the packets immutable as rewriting is either not possible because a) the relevant information is encrypted and inaccessible to the gateway or b) rewriting integrity-protected parts of the packet would invalidate the end-to-end integrity protection. There are essentially five approaches to handle such end-to-end

confidentiality and integrity protection while letting middleboxes access/modify data for different purposes:

1. Sharing credentials with middleboxes enables middleboxes to transform (e.g., decompress, convert, etc.) packets and re-apply the security measures after transformation. This method abandons end-to-end security and is only applicable to simple scenarios with a rudimentary security model.
2. Reusing the Internet wire format in the IoT makes conversion between IoT and Internet protocols unnecessary. However, it can lead to poor performance in some use cases because IoT specific optimizations (e.g., stateful or stateless compression) are not possible.
3. Selectively protecting vital and immutable packet parts with a MAC or with encryption requires a careful balance between performance and security. Otherwise, this approach will either result in poor performance (protect as much as possible) or poor security (compress and transform as much as possible).
4. Message authentication codes that sustain transformation can be realized by considering the order of transformation and protection (e.g., by creating a signature before compression so that the gateway can decompress the packet without recalculating the signature). [[ID-OSCOAP](#)] proposes a solution in this direction, also preventing proxies from changing relevant CoAP fields. Such an approach enables IoT specific optimizations but is more complex and may require application-specific transformations before security is applied. Moreover, it cannot be used with encrypted data because the lack of cleartext prevents gateways from transforming packets.
5. Object security based mechanisms can bridge the protocol worlds, but still requires that the two worlds use the same object security formats. Currently the IoT based object security format based on COSE [[ID-cose](#)] is different from the Internet based JOSE or CMS. Legacy devices on the Internet side will need to update to the newer IoT protocols to enable real end-to-end security. Furthermore, middleboxes do not have any access to the data and this approach does not prevent an attacker from modifying relevant fields in CoAP.

To the best of our knowledge, none of the mentioned security approaches that focus on the confidentiality and integrity of the communication exchange between two IP end-points provides a fully customizable solution in this problem space.

We note that end-to-end security could also be considered in the context of availability: making sure that the messages are delivered. In this case, the end-points cannot control this, but the middleboxes play a fundamental role to make sure that exchanged messages are not dropped, e.g., due to a DDoS attack.

7.1.4. New network architectures and paradigm

There is a multitude of new link layer protocols that aim to address the resource-constrained nature of IoT devices. For example, the IEEE 802.11 ah [[IEEE802ah](#)] has been specified for extended range and lower energy consumption to support Internet of Things (IoT) devices. Similarly, Low-Power Wide-Area Network (LPWAN) protocols such as LoRa [[lora](#)], Sigfox [[sigfox](#)], NarrowBand IoT (NB-IoT) [[nbiot](#)] are all designed for resource-constrained devices that require long range and low bit rates. While these protocols allow the IoT devices to conserve energy and operate efficiently, they also add additional security challenges. For example, the relatively small MTU can make security handshakes with large X509 certificates a significant overhead. At the same time, new communication paradigms also allow IoT devices to communicate directly amongst themselves with or without support from the network. This communication paradigm is also referred to as Device-to-Device (D2D) or Machine-to-Machine (M2M) or Thing-to-Thing (T2T) communication. D2D is primarily driven by network operators that want to utilize short range communication to improve the network performance and for supporting proximity based service

7.2. Bootstrapping of a Security Domain

Creating a security domain from a set of previously unassociated IoT devices is a key operation in the lifecycle of a thing and in the IoT network. This aspect is further elaborated and discussed in the T2TRG draft on bootstrapping [[ID-bootstrap](#)].

7.3. Operation

After the bootstrapping phase, the system enters the operational phase. During the operational phase, things can relate to the state information created during the bootstrapping phase in order to exchange information securely and in an authenticated fashion. In this section, we discuss aspects of communication patterns and network dynamics during this phase.

7.3.1. End-to-End Security

Providing end-to-end security is of great importance to address and secure individual T2T or H2T communication within one IoT domain. Moreover, end-to-end security associations are an important measure to bridge the gap between the IoT and the Internet. IKEv2, TLS and DTLS provide end-to-end security services including peer entity authentication, end-to-end encryption and integrity protection above the network layer and the transport layer respectively. Once bootstrapped, these functions can be carried out without online connections to third parties, making the protocols applicable for decentralized use in the IoT. However, protocol translation by intermediary nodes may invalidate end-to-end protection measures (see [Section 7.1.3](#)). Also these protocols require end-to-end connectivity between the devices and do not support store-and-forward scenarios. Object security is an option for such scenarios and the work on OSCOAP [[ID-OSCOAP](#)] is a potential solution in this space, in particular, in the context of forwarding proxies.

7.3.2. Group Membership and Security

In addition to end-to-end security, group key negotiation is an important security service for the T2Ts and Ts2T communication patterns in the IoT as efficient local broadcast and multicast relies on symmetric group keys.

All discussed protocols only cover unicast communication and therefore do not focus on group-key establishment. However, the Diffie-Hellman keys that are used in IKEv2 and HIP could be used for group Diffie-Hellman key-negotiations. Conceptually, solutions that provide secure group communication at the network layer (IPsec/IKEv2, HIP/Diet HIP) may have an advantage regarding the cryptographic overhead compared to application-focused security solutions (TLS/DTLS or OSCOAP). This is due to the fact that application-focused solutions require cryptographic operations per group application, whereas network layer approaches may allow to share secure group associations between multiple applications (e.g., for neighbor discovery and routing or service discovery). Hence, implementing shared features lower in the communication stack can avoid redundant security measures.

A number of group key solutions have been developed in the context of the IETF working group MSEC in the context of the MIKEY architecture [[WG-MSEC](#)][RFC4738]. These are specifically tailored for multicast and group broadcast applications in the Internet and should also be considered as candidate solutions for group key agreement in the IoT. The MIKEY architecture describes a coordinator entity that disseminates symmetric keys over pair-wise end-to-end secured

channels. However, such a centralized approach may not be applicable in a distributed environment, where the choice of one or several coordinators and the management of the group key is not trivial.

7.3.3. Mobility and IP Network Dynamics

It is expected that many things (e.g., wearable sensors, and user devices) will be mobile in the sense that they are attached to different networks during the lifetime of a security association. Built-in mobility signaling can greatly reduce the overhead of the cryptographic protocols because unnecessary and costly re-establishments of the session (possibly including handshake and key agreement) can be avoided. IKEv2 supports host mobility with the MOBIKE [[RFC4555](#)][[RFC4621](#)] extension. MOBIKE refrains from applying heavyweight cryptographic extensions for mobility. However, MOBIKE mandates the use of IPsec tunnel mode which requires to transmit an additional IP header in each packet. This additional overhead could be alleviated by using header compression methods or the Bound End-to-End Tunnel (BEET) mode [[ID-Nikander](#)], a hybrid of tunnel and transport mode with smaller packet headers.

HIP offers a simple yet effective mobility management by allowing hosts to signal changes to their associations [[RFC5206](#)]. However, slight adjustments might be necessary to reduce the cryptographic costs, for example, by making the public-key signatures in the mobility messages optional. Diet HIP does not define mobility yet but it is sufficiently similar to HIP to employ the same mechanisms. TLS and DTLS do not have standards for mobility support, however, work on DTLS mobility exists in the form of an Internet draft [[ID-Williams](#)]. The specific need for IP-layer mobility mainly depends on the scenario in which nodes operate. In many cases, mobility support by means of a mobile gateway may suffice to enable mobile IoT networks, such as body sensor networks. However, if individual things change their point of network attachment while communicating, mobility support may gain importance.

7.4. Software update

IoT devices have a reputation for being insecure at the time of manufacture. Yet they are often expected to stay functional in live deployments for years and even decades. Additionally, these devices typically operate unattended with direct Internet connectivity. Therefore, a remote software update mechanism to fix vulnerabilities, to update configuration settings, and for adding new functionality is needed.

Schneier [[SchneierSecurity](#)] in his essay expresses concerns about the status of software and firmware update mechanisms for Internet of

Things (IoT) devices. He highlights several challenges that hinder mechanisms for secure software update of IoT devices. First, there is a lack of incentives for manufactures, vendors and others on the supply chain to issue updates for their devices. Second, parts of the software running on the IoT devices is simply a binary blob without any source code available. Since the complete source code is not available, no patches can be written for that piece of code. Third, even when updates are available, users generally have to manually download and install those updates. However, users are never alerted about security updates and many times do not have the necessary expertise to manually administer the required updates.

The FTC staff report on Internet of Things - Privacy & Security in a Connected World [[FTCreport](#)] and the Article 29 Working Party Opinion 8/2014 on the on Recent Developments on the Internet of Things [[Article29](#)] also document the challenges for secure remote software update of IoT devices. They note that even providing such a software update capability may add new vulnerabilities for constrained devices. For example, a buffer overflow vulnerability in the implementation of a software update protocol (TR69) [[TR69](#)] and an expired certificate in a hub device [[wink](#)] demonstrate how the software update process itself can introduce vulnerabilities.

While powerful IoT devices that run general purpose operating systems can make use of sophisticated software update mechanisms known from the desktop world, a more considerate effort is needed for resource-constrained devices that don't have any operating system and are typically not equipped with a memory management unit or similar tools. The IAB also organized a workshop to understand the challenges for secure software update of IoT devices. A summary of the workshop and the proposed next steps have been documented [[iotsu](#)].

[7.5. Verifying device behavior](#)

Users often have a false sense of privacy when using new Internet of Things (IoT) appliances such as Internet-connected smart televisions, speakers and cameras. Recent revelations have shown that this user belief is often unfounded. Many IoT device vendors have been caught collecting sensitive private data through these connected appliances with or without appropriate user warnings [[cctv](#)].

An IoT device user/owner would like to monitor and know if the device is calling home (i.e. verify its operational behavior). The calling home feature may be necessary in some scenarios, such as during the initial configuration of the device. However, the user should be kept aware of the data that the device is sending back to the vendor.

For example, the user should be ensured that his/her TV is not sending data when he/she inserts a new USB stick.

Providing such information to the users in an understandable fashion is challenging. This is because the IoT devices are not only resource-constrained in terms of their computational capability, but also in terms of the user interface available. Also, the network infrastructure where these devices are deployed will vary significantly from one user environment to another. Therefore, where and how this monitoring feature is implemented still remains an open question.

7.6. End-of-life

Like all commercial devices, most IoT devices will be end-of-lifed by vendors or even network operators. This may be planned or unplanned (for example when the vendor or manufacturer goes bankrupt or when a network operator moves to a different type of networking technology). A user should still be able to use and perhaps even update the device. This requires for some form of authorization handover.

Although this may seem far fetched given the commercial interests and market dynamics, we have examples from the mobile world where the devices have been functional and up-to-date long after the original vendor stopped supporting the device. CyanogenMod for Android devices and OpenWrt for home routers are two such instances where users have been able to use and update their devices even after they were end-of-lifed. Admittedly these are not easy for an average users to install and configure on their devices. With the deployment of millions of IoT devices, simpler mechanisms are needed to allow users to add new root-of-trusts and install software and firmware from other sources once the device has been end-of-lifed.

7.7. Testing: bug hunting and vulnerabilities

Given that the IoT devices often have inadvertent vulnerabilities, both users and developers would want to perform extensive testing on their IoT devices, networks, and systems. Nonetheless, since the devices are resource-constrained and manufactured by multiple vendors, some of them very small, devices might be shipped with very limited testing, so that bugs can remain and can be exploited at a later stage. This leads to two main types of challenges:

1. It remains to be seen how the software testing and quality assurance mechanisms used from the desktop and mobile world will be applied to IoT devices to give end users the confidence that the purchased devices are robust.

2. It is also an open question how combination of devices of multiple vendors might actually lead to dangerous network configurations, e.g., if combination of specific devices can trigger unexpected behavior.

7.8. Quantum-resistance

Many IoT systems that are being deployed today will remain operational for many years. With the advancements made in the field of quantum computers, it is possible that large-scale quantum computers are available in the future for performing cryptanalysis on existing cryptographic algorithms and cipher suites. If this happened, it would have two consequences. First, functionalities enabled by means of RSA/ECC - namely key exchange, public-key encryption and signature - would not be secure anymore due to Shor's algorithm. Second, the security level of symmetric algorithms will decrease, e.g., the security of a block cipher with a key size of b bits will only offer $b/2$ bits of security due to Grover's algorithm.

This would require to move to quantum-resistant alternatives, in particular, for those functionalities involving key exchange, public-key encryption and signatures. While such future planning is hard, it may be a necessity in certain critical IoT deployments which are expected to last decades or more. Although increasing the key-size of the different algorithms is definitely an option, it would also incur additional computation overhead and network traffic. This would be undesirable in most scenarios. There have been recent advancements in quantum-resistant cryptography.

We refer to [[ETSI GR QSC 001](#)] for an extensive overview of existing quantum-resistant cryptography. [RFC7696](#) provides guidelines for cryptographic algorithm agility.

7.9. Privacy protection

Users will be surrounded by hundreds of connected devices. Even if the communication links are encrypted and protected, information about the users might be collected for different purposes affecting their privacy. In [[Ziegeldorf](#)], privacy in the IoT is defined as the threefold guarantee to the user for: 1. awareness of privacy risks imposed by smart things and services surrounding the data subject, 2. individual control over the collection and processing of personal information by the surrounding smart things, 3. awareness and control of subsequent use and dissemination of personal information by those entities to any entity outside the subject's personal control sphere.

Based on this definition, several privacy threats and challenges are identified in the work of Ziegeldorf:

1. Identification - refers to the identification of the users and their objects.
2. Localization - relates to the capability of locating a user and even tracking him.
3. Profiling - is about creating a profile of the user and her preferences.
4. Interaction - occurs when a user has been profiled and a given interaction is preferred, presenting (e.g., visually) some information that discloses private information.
5. Lifecycle transitions - take place when devices are, e.g., sold without properly removing private data.
6. Inventory attacks - happen if specific information about (smart) objects in possession of a user is disclosed.
7. Linkage - is about when information of two of more IoT systems is combined so that a broader view on the personal data is created.

When IoT systems are deployed, the above issues should be considered to ensure that private data remains private. How this is achieved remains an open issue.

7.10. Data leakage

IoT devices are resource constrained and often deployed in unattended environments or can just be bought in the Internet. Therefore, an attacker can have direct access to the device and apply more advance techniques that a traditional black box model does not consider such as side-channel attacks or code disassembly. By doing this, the attacker can try to retrieve data such as:

1. long term keys that might be used perform attacks on devices deployed in other locations.
2. source code that might let the user determine bugs or find exploits to perform other types of attacks, or just sell it,
3. proprietary algorithms that could be counterfeited or modified to perform advanced attacks.

Protection against such data leakage patterns is not trivial since devices are inherently resource-constrained. An open question is which techniques can be used to protect IoT devices in such a strong attack model.

7.11. Trustworthy IoT Operation

Flaws in the design and implementation of a secure IoT device and network can lead to secure vulnerabilities. An example is a flaw in the distribution of an Internet-connected IoT device in which a default password is used in all devices. Many IoT devices can be found in the Internet by means of tools such as Shodan, and if they have any vulnerability, it can then be exploited at scale, e.g., to launch DDoS attacks. This is not fiction but reality as Dyn, a major DNS was attacked by means of a DDoS attack originated from a large IoT botnet composed of thousands of compromised IP-cameras. Open questions in this area are:

1. How to prevent large scale vulnerabilities in IoT devices?
2. How to prevent attackers from exploiting vulnerabilities in IoT devices at large scale?
3. If the vulnerability has been exploited, how do we stop a large scale attack before any damage is caused?

Some ideas are being explored to address this issue. One of these approaches refers to the specification of Manufacturer Usage Description (MUD) files [[ID-MUD](#)]. The idea behind MUD files is simple: devices would disclose the location of its MUD file to the network during installation. The network can then (i) retrieve those files, (ii) learn from the manufacturers the intended usage of the devices, e.g., which services they require to access, and then (iii) create suitable filters.

8. Conclusions and Next Steps

This Internet Draft provides IoT security researchers, system designers and implementers with an overview of both operational and security requirements in the IP-based Internet of Things. We discuss a general threat model, security issues, state of the art, and tools to mitigate security threats. We further introduce a number of potential security profiles fitting different types of IoT deployments and discuss key security challenges.

Although big steps have been realized during the last years, as summarized in Section [Section 5.1](#) and many organizations are publishing general recommendations (Section [Section 5.3](#)) describing how the IoT should be secured, there are many challenges ahead that require further attention. Challenges of particular importance are bootstrapping of security, group security, secure software updates, long-term security and quantum-resistance, privacy protection, data leakage prevention - where data could be cryptographic keys, personal

data, or even algorithms - and ensuring a trustworthy IoT operation. All these problems are important; however, different deployment environments have different operational and security demands. Thus, a potential approach is the definition and standardization of security profiles, e.g., similar to the exemplary profiles introduced in Section [Section 6](#), to ensure minimum security capabilities in different environments while ensuring interoperability.

9. Security Considerations

This document reflects upon the requirements and challenges of the security architectural framework for the Internet of Things.

10. IANA Considerations

This document contains no request to IANA.

11. Acknowledgments

We gratefully acknowledge feedback and fruitful discussion with Tobias Heer, Robert Moskowitz, and Thorsten Dahm. We acknowledge the additional authors of the previous version of this document Sye Loong Keoh, Rene Hummen and Rene Struik.

12. Informative References

[Article29]

"Opinion 8/2014 on the on Recent Developments on the Internet of Things", Web http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2014/wp223_en.pdf, n.d..

[AUTO-ID] "AUTO-ID LABS", Web <http://www.autoidlabs.org/>, September 2010.

[BACNET] "BACnet", Web <http://www.bacnet.org/>, February 2011.

[BITAG] "Internet of Things (IoT) Security and Privacy Recommendations", Web <http://www.bitag.org/report-internet-of-things-security-privacy-recommendations.php>, n.d..

[cctv] "Backdoor In MvPower DVR Firmware Sends CCTV Stills To an Email Address In China", Web <https://hardware.slashdot.org/story/16/02/17/0422259/backdoor-in-mvpower-dvr-firmware-sends-cctv-stills-to-an-email-address-in-china>, n.d..

- [CSA] "Security Guidance for Early Adopters of the Internet of Things (IoT)", Web https://downloads.cloudsecurityalliance.org/whitepapers/Security_Guidance_for_Early_Adopters_of_the_Internet_of_Things.pdf, n.d..
- [d2dsecurity]
Haus, M., Waqas, M., Ding, A., Li, Y., Tarkoma, S., and J. Ott, "Security and Privacy in Device-to-Device (D2D) Communication: A Review", Paper IEEE Communications Surveys and Tutorials, 2016.
- [DALI] "DALI", Web <http://www.dalibydesign.us/dali.html>, February 2011.
- [DHS] "Strategic Principles For Securing the Internet of Things (IoT)", Web https://www.dhs.gov/sites/default/files/publications/Strategic_Principles_for_Securing_the_Internet_of_Things-2016-1115-FINAL....pdf, n.d..
- [ENISA_ICS]
"Communication network dependencies for ICS/SCADA Systems", European Union Agency For Network And Information Security , February 2017.
- [ETSI_GR_QSC_001]
"Quantum-Safe Cryptography (QSC);Quantum-safe algorithmic framework", European Telecommunications Standards Institute (ETSI) , June 2016.
- [Fairhair]
"Fairhair Alliance", Web <https://www.fairhair-alliance.org/>, n.d..
- [FCC] "Federal Communications Commission Response 12-05-2016", FCC , February 2016.
- [FTCreport]
"FTC Report on Internet of Things Urges Companies to Adopt Best Practices to Address Consumer Privacy and Security Risks", Web <https://www.ftc.gov/news-events/press-releases/2015/01/ftc-report-internet-things-urges-companies-adopt-best-practices>, n.d..

[GSMASecurity]

"GSMA IoT Security Guidelines", Web
<http://www.gsma.com/connectedliving/future-iot-networks/iot-security-guidelines/>, n.d..

[ID-6lodect]

Mariager, P., Petersen, J., Shelby, Z., Logt, M., and D. Barthel, "Transmission of IPv6 Packets over DECT Ultra Low Energy", [draft-ietf-6lo-dect-ule-09](#) , December 2016.

[ID-6lonfc]

Choi, Y., Hong, Y., Youn, J., Kim, D., and J. Choi, "Transmission of IPv6 Packets over Near Field Communication", [draft-ietf-6lo-nfc-05](#) , October 2016.

[ID-6tisch]

Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", [draft-ietf-6tisch-architecture-11](#) , January 2017.

[ID-aceoauth]

Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE)", [draft-ietf-ace-oauth-authz-05](#) , March 2011.

[ID-bootstrap]

Sarikaya, B. and M. Sethi, "Secure IoT Bootstrapping : A Survey", [draft-sarikaya-t2trg-sbootstrapping-01](#) , July 2016.

[ID-cose]

Schaad, J., "CBOR Object Signing and Encryption (COSE)", [draft-ietf-cose-msg-24](#) , November 2016.

[ID-Daniel]

Park, S., Kim, K., Haddad, W., Chakrabarti, S., and J. Laganier, "IPv6 over Low Power WPAN Security Analysis", [draft-daniel-6lowpan-security-analysis-05](#) , March 2011.

[ID-dietesp]

Migault, D., Guggemos, T., and C. Bormann, "Diet-ESP: a flexible and compressed format for IPsec/ESP", [draft-mglt-6lo-diet-esp-02](#) , August 2016.

[ID-Hartke]

Hartke, K. and O. Bergmann, "Datagram Transport Layer Security in Constrained Environments", [draft-hartke-core-codtls-02](#) , July 2012.

- [ID-HIP] Moskowitz, R., "HIP Diet EXchange (DEX)", [draft-moskowitz-hip-rg-dex-06](#) , May 2012.
- [ID-Moore] Moore, K., Barnes, R., and H. Tschofenig, "Best Current Practices for Securing Internet of Things (IoT) Devices", [draft-moore-iot-security-bcp-00](#) , October 2016.
- [ID-MUD] Lear, E., Droms, R., and D. Domascanu, "Manufacturer Usage Description Specification", March 2017.
- [ID-Nikander] Nikander, P. and J. Melen, "A Bound End-to-End Tunnel(BEET) mode for ESP", [draft-nikander-esp-beet-mode-09](#) , August 2008.
- [ID-OFlynn] O'Flynn, C., Sarikaya, B., Ohba, Y., Cao, Z., and R. Cragie, "Security Bootstrapping of Resource-Constrained Devices", [draft-oflynn-core-bootstrapping-03](#) , November 2010.
- [ID-OSCOAP] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security of CoAP (OSCOAP)", [draft-selander-ace-object-security-05](#) , July 2016.
- [ID-proHTTPCoAP] Castellani, A., Loreto, S., Rahman, A., Fossati, T., and E. Dijk, "Best practices for HTTP-CoAP mapping implementation", [draft-castellani-core-http-mapping-07](#) , February 2013.
- [ID-rd] Shelby, Z., Koster, M., Bormann, C., and P. Stok, "CoRE Resource Directory", [draft-ietf-core-resource-directory-09](#) , October 2016.
- [ID-senml] Jennings, C., Shelby, Z., Arkko, J., Keranen, A., and C. Bormann, "Media Types for Sensor Measurement Lists (SenML)", [draft-ietf-core-resource-directory-09](#) , October 2016.
- [ID-Tsao] Tsao, T., Alexander, R., Dohler, M., Daza, V., and A. Lozano, "A Security Framework for Routing over Low Power and Lossy Networks", [draft-ietf-roll-security-framework-07](#) , January 2012.

[ID-Williams]

Williams, M. and J. Barrett, "Mobile DTLS", [draft-barrett-mobile-dtls-00](#) , March 2009.

[IEEE802ah]

"Status of Project IEEE 802.11ah, IEEE P802.11- Task Group AH-Meeting Update.",
Web http://www.ieee802.org/11/Reports/tgah_update.htm,
n.d..

[IIoT]

"Industrial Internet Consortium",
Web <http://www.iiconsortium.org/>, n.d..

[IoTSecFoundation]

"Establishing Principles for Internet of Things Security",
Web <https://iotsecurityfoundation.org/establishing-principles-for-internet-of-things-security/>, n.d..

[iotsu]

"Patching the Internet of Things: IoT Software Update Workshop 2016", Web
<https://www.ietf.org/blog/2016/07/patching-the-internet-of-things-iot-software-update-workshop-2016/>, n.d..

[IPSO]

"IPSO Alliance", Web <http://www.ipso-alliance.org>, n.d..

[JOURNAL-Perrig]

Perrig, A., Szewczyk, R., Wen, V., Culler, D., and J. Tygar, "SPINS: Security protocols for Sensor Networks",
Journal Wireless Networks, September 2002.

[loral]

"LoRa - Wide Area Networks for IoT", Web <https://www.lora-alliance.org/>, n.d..

[nbiot]

"NarrowBand IoT", Web
http://www.3gpp.org/ftp/tsg_ran/TSG_RAN/TSGR_69/Docs/RP-151621.zip, n.d..

[NHTSA]

"Cybersecurity Best Practices for Modern Vehicles", Web
https://www.nhtsa.gov/staticfiles/nvs/pdf/812333_CybersecurityForModernVehicles.pdf, n.d..

[NIST]

Dworkin, M., "NIST Specification Publication 800-38B",
2005.

[NIST-Guide]

Ross, R., McEVILLEY, M., and J. Oren, "Systems Security Engineering", Web <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160.pdf>, n.d..

[nist_lightweight_project]

"NIST lightweight Project", Web www.nist.gov/programs-projects/lightweight-cryptography, www.nist.gov/sites/default/files/documents/2016/10/17/sonmez-turan-presentation-lwc2016.pdf, n.d..

[OCF]

"Open Connectivity Foundation", Web <https://openconnectivity.org/>, n.d..

[OneM2M]

"OneM2M", Web <http://www.onem2m.org/>, n.d..

[OWASP]

"IoT Security Guidance", Web https://www.owasp.org/index.php/IoT_Security_Guidance, n.d..

[PROC-Chan]

Chan, H., Perrig, A., and D. Song, "Random Key Predistribution Schemes for Sensor Networks", Proceedings IEEE Symposium on Security and Privacy, 2003.

[PROC-Gupta]

Gupta, V., Wurm, M., Zhu, Y., Millard, M., Fung, S., Gura, N., Eberle, H., and S. Shantz, "Sizzle: A Standards-based End-to-End Security Architecture for the Embedded Internet", Proceedings Pervasive Computing and Communications (PerCom), 2005.

[PROC-Smetters-02]

Balfanz, D., Smetters, D., Steward, P., and H. Chi Wong,, "Talking To Strangers: Authentication in Ad-Hoc Wireless Networks", Paper NDSS, 2002.

[PROC-Smetters-04]

Balfanz, D., Durfee, G., Grinter, R., Smetters, D., and P. Steward, "Network-in-a-Box: How to Set Up a Secure Wireless Network in Under a Minute", Paper USENIX, 2004.

[PROC-Stajano-99]

Stajano, F. and R. Anderson, "Resurrecting Duckling - Security Issues for Adhoc Wireless Networks", 7th International Workshop Proceedings, November 1999.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", [RFC 2818](#), DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", [RFC 3261](#), DOI 10.17487/RFC3261, June 2002, <<http://www.rfc-editor.org/info/rfc3261>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", [RFC 3748](#), DOI 10.17487/RFC3748, June 2004, <<http://www.rfc-editor.org/info/rfc3748>>.
- [RFC3756] Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", [RFC 3756](#), DOI 10.17487/RFC3756, May 2004, <<http://www.rfc-editor.org/info/rfc3756>>.
- [RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", [RFC 3833](#), DOI 10.17487/RFC3833, August 2004, <<http://www.rfc-editor.org/info/rfc3833>>.
- [RFC4016] Parthasarathy, M., "Protocol for Carrying Authentication and Network Access (PANA) Threat Analysis and Security Requirements", [RFC 4016](#), DOI 10.17487/RFC4016, March 2005, <<http://www.rfc-editor.org/info/rfc4016>>.
- [RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", [RFC 4251](#), DOI 10.17487/RFC4251, January 2006, <<http://www.rfc-editor.org/info/rfc4251>>.
- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", [RFC 4555](#), DOI 10.17487/RFC4555, June 2006, <<http://www.rfc-editor.org/info/rfc4555>>.
- [RFC4621] Kivinen, T. and H. Tschofenig, "Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol", [RFC 4621](#), DOI 10.17487/RFC4621, August 2006, <<http://www.rfc-editor.org/info/rfc4621>>.

- [RFC4738] Ignjatic, D., Dondeti, L., Audet, F., and P. Lin, "MIKEY-RSA-R: An Additional Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)", [RFC 4738](#), DOI 10.17487/RFC4738, November 2006, <<http://www.rfc-editor.org/info/rfc4738>>.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", [RFC 4919](#), DOI 10.17487/RFC4919, August 2007, <<http://www.rfc-editor.org/info/rfc4919>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", [RFC 4944](#), DOI 10.17487/RFC4944, September 2007, <<http://www.rfc-editor.org/info/rfc4944>>.
- [RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", [RFC 5191](#), DOI 10.17487/RFC5191, May 2008, <<http://www.rfc-editor.org/info/rfc5191>>.
- [RFC5206] Nikander, P., Henderson, T., Ed., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", [RFC 5206](#), DOI 10.17487/RFC5206, April 2008, <<http://www.rfc-editor.org/info/rfc5206>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5713] Moustafa, H., Tschofenig, H., and S. De Cnodder, "Security Threats and Security Requirements for the Access Node Control Protocol (ANCP)", [RFC 5713](#), DOI 10.17487/RFC5713, January 2010, <<http://www.rfc-editor.org/info/rfc5713>>.
- [RFC5903] Fu, D. and J. Solinas, "Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2", [RFC 5903](#), DOI 10.17487/RFC5903, June 2010, <<http://www.rfc-editor.org/info/rfc5903>>.
- [RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., Ed., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA) Relay Element", [RFC 6345](#), DOI 10.17487/RFC6345, August 2011, <<http://www.rfc-editor.org/info/rfc6345>>.

- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](#), DOI 10.17487/RFC6347, January 2012, <<http://www.rfc-editor.org/info/rfc6347>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", [RFC 6550](#), DOI 10.17487/RFC6550, March 2012, <<http://www.rfc-editor.org/info/rfc6550>>.
- [RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", [RFC 6551](#), DOI 10.17487/RFC6551, March 2012, <<http://www.rfc-editor.org/info/rfc6551>>.
- [RFC6568] Kim, E., Kaspar, D., and JP. Vasseur, "Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", [RFC 6568](#), DOI 10.17487/RFC6568, April 2012, <<http://www.rfc-editor.org/info/rfc6568>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", [RFC 6690](#), DOI 10.17487/RFC6690, August 2012, <<http://www.rfc-editor.org/info/rfc6690>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), DOI 10.17487/RFC6749, October 2012, <<http://www.rfc-editor.org/info/rfc6749>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", [RFC 7049](#), DOI 10.17487/RFC7049, October 2013, <<http://www.rfc-editor.org/info/rfc7049>>.
- [RFC7158] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", [RFC 7158](#), DOI 10.17487/RFC7158, March 2014, <<http://www.rfc-editor.org/info/rfc7158>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", [RFC 7252](#), DOI 10.17487/RFC7252, June 2014, <<http://www.rfc-editor.org/info/rfc7252>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, [RFC 7296](#), DOI 10.17487/RFC7296, October 2014, <<http://www.rfc-editor.org/info/rfc7296>>.

- [RFC7390] Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for the Constrained Application Protocol (CoAP)", [RFC 7390](#), DOI 10.17487/RFC7390, October 2014, <<http://www.rfc-editor.org/info/rfc7390>>.
- [RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", [RFC 7401](#), DOI 10.17487/RFC7401, April 2015, <<http://www.rfc-editor.org/info/rfc7401>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", [RFC 7515](#), DOI 10.17487/RFC7515, May 2015, <<http://www.rfc-editor.org/info/rfc7515>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", [RFC 7517](#), DOI 10.17487/RFC7517, May 2015, <<http://www.rfc-editor.org/info/rfc7517>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", [RFC 7519](#), DOI 10.17487/RFC7519, May 2015, <<http://www.rfc-editor.org/info/rfc7519>>.
- [RFC7668] Nieminen, J., Savolainen, T., Isomaki, M., Patil, B., Shelby, Z., and C. Gomez, "IPv6 over BLUETOOTH(R) Low Energy", [RFC 7668](#), DOI 10.17487/RFC7668, October 2015, <<http://www.rfc-editor.org/info/rfc7668>>.
- [RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", [BCP 201](#), [RFC 7696](#), DOI 10.17487/RFC7696, November 2015, <<http://www.rfc-editor.org/info/rfc7696>>.
- [RFC7815] Kivinen, T., "Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation", [RFC 7815](#), DOI 10.17487/RFC7815, March 2016, <<http://www.rfc-editor.org/info/rfc7815>>.
- [RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", [RFC 7925](#), DOI 10.17487/RFC7925, July 2016, <<http://www.rfc-editor.org/info/rfc7925>>.
- [RG-T2TRG] "IRTF Thing-to-Thing (T2TRG) Research Group", Web <https://datatracker.ietf.org/rg/t2trg/charter/>, December 2015.

[SchneierSecurity]

"The Internet of Things Is Wildly Insecure--And Often Unpatchable", Web

https://www.schneier.com/essays/archives/2014/01/the_internet_of_thin.html, n.d..

[sigfox] "Sigfox - The Global Communications Service Provider for the Internet of Things (IoT)",

Web <https://www.sigfox.com/>, n.d..

[SPEKE] "IEEE P1363.2: Password-based Cryptography", 2008.

[THESIS-Langheinrich]

Langheinrich, M., "Personal Privacy in Ubiquitous Computing", PhD Thesis ETH Zurich, 2005.

[Thread] "Thread Group", Web <http://threadgroup.org/>, n.d..

[TinyDTLS]

"TinyDTLS", Web <http://tinydtls.sourceforge.net/>, February 2012.

[TR69] "Too Many Cooks - Exploiting the Internet-of-TR-069-Things", Web https://media.ccc.de/v/31c3_-_6166_-_en_-_saal_6_-_201412282145_-_too_many_cooks_-_exploiting_the_internet-of-tr-069-things_-_lior_oppenheim_-_shahar_tal, n.d..

[WG-6LoWPAN]

"IETF 6LoWPAN Working Group",
Web <http://tools.ietf.org/wg/6lowpan/>, February 2011.

[WG-ACE] "IETF Authentication and Authorization for Constrained Environments (ACE) Working Group",

Web <https://datatracker.ietf.org/wg/ace/charter/>, June 2014.

[WG-CoRE] "IETF Constrained RESTful Environment (CoRE) Working Group", Web <https://datatracker.ietf.org/wg/core/charter/>, February 2011.

[WG-LWIG] "IETF Light-Weight Implementation Guidance (LWIG) Working Group", Web <https://datatracker.ietf.org/wg/lwig/charter/>, March 2011.

[WG-MSEC] "MSEC Working Group",

Web <http://datatracker.ietf.org/wg/msec/>, n.d..

[wink] "Wink's Outage Shows Us How Frustrating Smart Homes Could Be",
Web <http://www.wired.com/2015/04/smart-home-headaches/>,
n.d..

[ZB] "ZigBee Alliance", Web <http://www.zigbee.org/>, February 2011.

[Ziegeldorf]
Ziegeldorf, J., Garcia-Morchon, O., and K. Wehrle,,
"Privacy in the Internet of Things: Threats and Challenges", Paper Security and Communication Networks - Special Issue on Security in a Completely Interconnected World, 2013.

Authors' Addresses

Oscar Garcia-Morchon
Philips IP&S
High Tech Campus 5
Eindhoven, 5656 AA
The Netherlands

Email: oscar.garcia-morchon@philips.com

Sandeep S. Kumar
Philips Research
High Tech Campus
Eindhoven, 5656 AA
The Netherlands

Email: sandeep.kumar@philips.com

Mohit Sethi
Ericsson
Hirsalantie 11
Jorvas
Finland

Email: mohit@piuha.net

