Security Challenges in the IP-based Internet of Things*

Tobias Heer*, Oscar Garcia-Morchon†, René Hummen*,
Sye Loong Keoh†, Sandeep S. Kumar†, and Klaus Wehrle*

*COMSYS Group, RWTH Aachen University, Germany
and †Philips Research, the Netherlands
{heer,hummen,wehrle}@cs.rwth-aachen.de,
{oscar.garcia,sye.loong.keoh,sandeep.kumar}@philips.com

Abstract. A direct interpretation of the term *Internet of Things* refers to the use of standard Internet protocols for the human-to-thing or thing-to-thing communication in embedded networks. Although the security needs are well-recognized in this domain, it is still not fully understood how existing IP security protocols and architectures can be deployed. In this paper, we discuss the applicability and limitations of existing Internet protocols and security architectures in the context of the Internet of Things. First, we give an overview of the deployment model and general security needs. We then present challenges and requirements for IP-based security solutions and highlight specific technical limitations of standard IP security protocols.

Keywords: Security, Internet of Things, IETF

1 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 [1] 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 use of standard Internet technology in the IoT.

The introduction of IPv6 and web services as fundamental building blocks for IoT applications [2] 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) a 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

*This article appeared in the Springer Journal on Wireless Personal Communications. The final publication is available at http://www.springerlink.com/content/dg866877x7187765/

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 paper 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. Our discussion shows that, although current standardization efforts are making progress in pursuing the secure IP-based IoT, security remains to date, at least partially, unsolved.

The rest of the paper is organized as follows. Section 2 depicts the lifecycle of a *thing* and gives general definitions for the main security aspects within the IoT domain. In Section 3, we review existing protocols and work done in the area of security for wireless sensor networks. Section 4 identifies general challenges and needs for an IoT security protocol design and discusses existing protocols and protocol proposals against the identified requirements. Finally, Section 5 concludes the paper.

2 The *Thing* Lifecycle and Architectural Considerations

We consider the installation of a Building Automation Control (BAC) system to illustrate the lifecycle of a *thing*. A BAC system consists of a network of interconnected nodes that perform 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 a single manufacturer creates 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 run the functions of the BAC system. During this operational phase, the device is under the control of the system owner. For devices with lifetimes that span 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. 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 network under a different owner by starting the lifecycle over again. Figure 1 shows the generic lifecycle of a *thing*. This generic lifecycle is also applicable for IoT scenarios other than BAC systems.

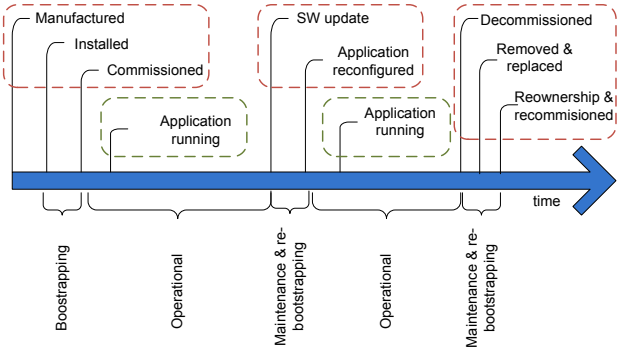

Fig. 1. The lifecycle of a device in the Internet of Things

At present, BAC systems use legacy building control standards such as BACNet [3] or DALI [4] with independent networks for each subsystem (HVAC, lighting, etc.). However, this separation of functionality adds further complexity and costs to the configuration and maintenance of the different networks within the same building. As a result, more recent building control networks employ IP-based standards allowing seamless control over the various nodes with a single management system. While allowing for easier integration, this shift towards IP-based standards results in new requirements regarding the implementation of IP security protocols on constrained devices and the bootstrapping of security keys for devices across multiple manufacturers.

2.1 Security Aspects

The term security subsumes a wide range of different concepts. In the first place, it refers to the basic provision of security services including confidentiality, authentication, integrity, authorization, non-repudiation, and availability. These security services can be implemented by means of different cryptographic mechanisms, such as block ciphers, hash functions, or signature algorithms. For each of these mechanisms, a solid key management infrastructure is fundamental to handling the required cryptographic keys.

In the context of the IoT, however, security must not only focus on the required security services, but also on how these are realized in the overall system

and how the security functionalities are executed. To this end, we use the following terminology to analyze and classify security aspects in the IoT:

- The *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*.
- The *security model of a node* describes how the security parameters, processes, and applications are managed in a *thing*. This includes aspects such as process separation, secure storage of keying materials, etc.
- *Security bootstrapping* denotes the process by which a *thing* securely joins the IoT at a given location and point in time. Bootstrapping includes the authentication and authorization of a device as well as the transfer of security parameters allowing for trusted operation.
- *Network security* describes the mechanisms applied within a network to ensure trusted operation of the IoT. Specifically, it prevents attackers from endangering or modifying the expected operation of networked *things*. Network security can include a number of mechanisms ranging from secure routing to data link layer and network layer security.
- *Application security* guarantees that only trusted instances of an application running in the IoT can communicate with each other, while illegitimate instances cannot interfere.

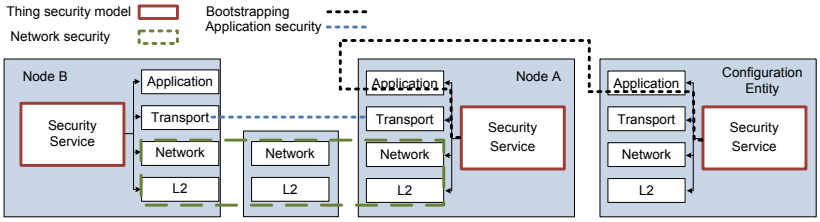

Fig. 2. Overview of Security Mechanisms

We now discuss an exemplary security architecture relying on a *configuration entity* for the management of the system with regard to the introduced security aspects (see Figure 2). This example illustrates how different security concepts and the lifecycle phases map to the Internet communication stack. Assume a centralized architecture in which a configuration entity stores and manages the identities of the *things* associated with the 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 the keying material. The security service on a *thing* in turn stores the received keying material for the network layer and application security mechanisms to resort to for secure communication. *Things* can then securely communicate with each other during their operational phase by means of the deployed network and application security mechanisms.

3 State of the Art

Nowadays, there exists a multitude of control protocols for the IoT. For BAC systems, the ZigBee standard [5], BACNet [3], or DALI [4] play key roles. Recent trends, however, focus on an all-IP approach for system control. Currently, a number of IETF working groups are designing new protocols for resource constrained networks of smart *things*. The 6LoWPAN working group [6] focuses on the definition of methods and protocols for the efficient transmission and adaptation of IPv6 packets over IEEE 802.15.4 networks [7]. The CoRE working group [8] 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) [9], that runs over UDP and enables efficient application-level communication for *things*.

3.1 IP-based security solutions

In the context of the IP-based IoT solutions, consideration of TCP/IP security protocols is important as these protocols are designed to fit the IP network ideology and technology. While a wide range of specialized as well as general-purpose key exchange and security solutions exist for the Internet domain, we focus on the discussion of IKEv2/IPsec [10], TLS/SSL [11], DTLS [12], HIP [13][14], PANA [15], and EAP [16] in this paper. Many of these protocols are currently discussed as candidate solutions in the 6LoWPAN and CoRE IETF working groups. Application layer solutions such as SSH [17] also exist, however, these are currently not considered. Figure 3 depicts the relationships between the discussed protocols.

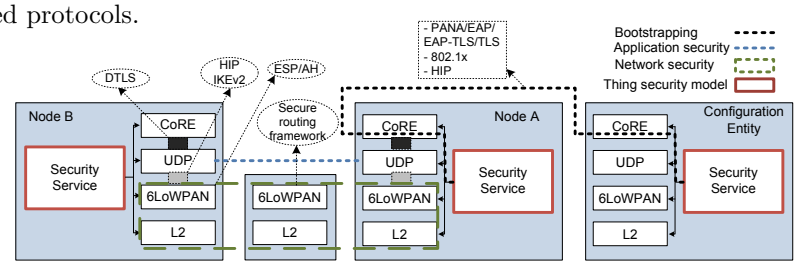

Fig. 3. Relationships between IP-based security protocols

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 [18] 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.

3.2 Wireless Sensor Network Security and Beyond

A variety of key agreement and privacy protection protocols that are tailored to IoT scenarios have been introduced in the literature. For instance, random key pre-distribution schemes [19] or more centralized solutions, such as SPINS [20], have been proposed for key establishment in wireless sensor networks. The ZigBee standard [5] for sensor networks defines a security architecture based on an online trust center that is in charge of handling the security relationships within a ZigBee network. Personal privacy in ubiquitous computing has been studied extensively, e.g., in [21]. Due to resource constraints and the specialization to meet specific requirements, these solutions often implement a collapsed cross-layer optimized communication stack (e.g., without task-specific network layers and layered packet headers). Consequently, they cannot directly be adapted to the requirements of the Internet due to the nature of their design.

Despite important steps done by, e.g., Gupta et al. [22], to show the feasibility of an end-to-end standard security architecture for the embedded Internet, the Internet and the IoT domain still do not fit together easily. This is mainly due to the fact that IoT security solutions are often tailored to the specific scenario requirements without considering interoperability with Internet protocols. On the other hand, the direct use of existing Internet security protocols in the IoT might lead to inefficient or insecure operation as we show in our discussion below.

4 Challenges for a Secure Internet of Things

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*. Table 4 summarizes which requirements need to be met in the lifecycle phases as well as the considered protocols. The structure of this section follows the structure of the table. 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.

4.1 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.

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. Especially, fate-sharing of packets in flight, as implemented by DTLS, aggravate the resulting performance loss.

Scarce CPU and memory resources limit the use of resource-demanding cryptoprimitives, such as public-key cryptography as used in most Internet security standards. This is especially true, if the basic cryptoblocks need to be used frequently or if 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 [23][24][14][18]. Moreover, all protocols have been revised in the last years to enable crypto agility, making cryptographic primitives interchangeable. Diet HIP takes the reduction of the cryptographic load one step further by focusing on cryptographic primitives that are to be expected to be enabled in hardware on IEEE 802.15.4 compliant devices. For example, Diet HIP does not require cryptographic hash functions but uses a CMAC [25] based mechanism, which can directly use the AES hardware available in standard sensor platforms. However, these improvements are only a first step in reducing the

	Bootstrapping phase	**Operational phase**
Requirements	Incremental deployment Identity and Key establishment Privacy-aware identification Group creation	End-to-end security Mobility support Group membership management
Protocols	IKEv2 TLS / DTLS HIP / Diet-HIP PANA/EAP	IKEv2/MOBIKE TLS / DTLS HIP / Diet-HIP

Table 1. Challenges and protocols for secure IoT

computation and communication overhead of Internet protocols. The question remains if other approaches can be applied to reduce the cost of 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, judgments on the energy consumption of a particular protocol cannot be made without tailor-made IoT implementations.

DoS 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 defenses 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 defense 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 misconfigured or malfunctioning *things*.

Protocol Translation and End-to-End Security 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 pendants 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 gateways, they become major obstacles if end-to-end security measures between IoT devices and Internet hosts are used.

Cryptographic payload processing applies message authentication codes or encryption to packets. 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 four solutions for this problem:

- *Sharing symmetric keys with gateways* enables gateways to transform (e.g., de-compress, 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.
- *Reusing the Internet wire format in the IoT* makes conversion between IoT and Internet protocols unnecessary. However, it leads to poor performance because IoT specific optimizations (e.g., stateful or stateless compression) are not possible.
- *Selectively protecting vital and immutable packet parts* with a message authentication code 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).
- *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). This 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.

To the best of our knowledge, none of the mentioned security protocols provides a fully customizable solution in this problem space. In fact, all discussed protocols usually provide end-to-end secured connection that do not afford translation at a gateway. An exception is the usage of PANA and EAP since (i) they allow for a number of configurations regarding the location of, e.g., the EAP authenticator and authentication server and (ii) the layered architecture might allow for authentication at different places. The drawback of this approach, however, lies in its high signaling traffic volume compared to other approaches. Hence, future work is required to ensure security, performance *and* interoperability between IoT and the Internet.

4.2 Bootstrapping of a Security Domain

Creating a security domain from a set of previously unassociated IoT devices is another important operation in the lifecycle of a *thing* and in the IoT network. In this section, we discuss general forms of network operation, how to communicate a *thing's* identity and the privacy implications arising from the communication of this identity.

Distributed vs. Centralized Architecture and Operation 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.

Decentralized architectures, on the other hand, allow to create ad-hoc security domains that might not require an online management entity and are operative in a 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.

Bootstrapping a *thing*'s identity and keying materials Bootstrapping refers to the process by which a device is associated to another one, to a network, or to a system. The way it is performed depends upon the architecture: centralized or distributed.

In a *distributed approach*, a Diffie-Hellman type of handshake can allow two peers to agree on a common secret. In general, IKEv2, HIP, TLS, DTLS, can perform key exchanges and the setup of security associations without online connections to a trust center. If we do not consider the resource limitations of *things*, certificates and certificate chains can be employed to securely communicate capabilities in such a decentralized scenario (e.g., for IKEv2, TLS, and DTLS). HIP and Diet HIP do not directly use certificates for identifying a host, however certificate handling capabilities exist for HIP and the same protocol logic could be used for Diet HIP. It is noteworthy, that Diet HIP does not *require* a *thing* to implement cryptographic hashes. Hence, some lightweight implementations of Diet HIP might not be able to verify certificates unless a hash function is implemented by the thing.

An additional complicating factor may be the administrative structure of the network. If several administrative entities (e.g., multiple network operators) bootstrap and manage the network, common security anchors must be found to achieve interoperability between devices in terms of security. Defining certificate

hierarchies via certificate chains can model more complex relations between security domains and the devices belonging to these. However, the tight packet size limitations in the IoT domain must be considered. Hence, using chains of certificates may require secure support for packet fragmentation and reassembly.

In a centralized architecture, preconfigured keys or certificates held by a *thing* can be used for the distribution of operational keys in a given security domain. A current proposal [26] refers to the use of PANA for the transport of EAP messages between the PANA client (the joining *thing*) and the PANA Authentication Agent (PAA), the 6LBR. EAP is thereby used to authenticate the identity of the joining *thing*. After the successful authentication, the PANA PAA provides the joining *thing* with fresh network and security parameters.

IKEv2, HIP, TLS, and DTLS could be applied as well for the transfer of configuration parameters in a centralized scenario. While HIP's cryptographic secret identifies the *thing*, the other protocols do not represent primary identifiers but are used instead to bind other identifiers such as the operation keys to the public-key identities.

In addition to the protocols, operational aspects during bootstrapping are of key importance as well. Many standard Internet protocols assume that the identity of a host is either available by using secondary services like certificate authorities or secure name resolution (e.g., DNSsec) or can be provided over a side channel (entering passwords via screen and keyboard). While these assumptions may hold in traditional networks, intermittent connectivity, localized communication, and lack of input methods complicate the situation for the IoT.

The order in which the *things* within a security domain are bootstrapped plays an important role as well. In [27], the PANA relay element is introduced, relaying PANA messages between a PaC (joining *thing*) and PAA of a segment [26]. This approach forces *circular commissioning*, i.e., *things* can only be bootstrapped in circles starting from those closer to the PAA. Although this can work a priori, it imposes important limitations in actual use cases in which an installer without technical background has to roll-out the system.

Privacy-aware identification During the last years, the introduction of RFID tags has raised privacy concerns because anyone might access and track tags. As the IoT involves not only passive devices, but also includes active and sensing devices, the IoT might irrupt even deeper in people's privacy spheres. Thus, IoT protocols should be designed to avoid these privacy threats during bootstrapping and operation where deemed necessary. In H2T and T2T interactions, privacy-aware identifiers might be used to prevent unauthorized user tracking. Similarly, authentication can be used to prove membership of a group without revealing unnecessary individual information.

TLS and DTLS provide the option of only authenticating the responding host. This way, the initiating host can stay anonymous. If authentication for the initiating host is required as well, either public-key certificates or authentication via the established encrypted payload channel can be employed. Such a setup allows to only reveal the responder's identity to possible eavesdroppers.

HIP and IKEv2 use public-key identities to authenticate the initiator of a connection. These identities could easily be traced if no additional protection were in place. IKEv2 transmits this information in an encrypted packet. Likewise, HIP provides the option to keep the identity of the initiator secret from eavesdroppers by encrypting it with the symmetric key generated during the handshake. However, Diet HIP cannot provide a similar feature because the identity of the initiator simultaneously serves as static Diffie-Hellman key. Note that all discussed solutions could use anonymous public-key identities that change for each communication. However, such identity cycling may require a considerable computational effort for generating new asymmetric key pairs. In addition to the built-in privacy features of the protocols discussed here, a large body of anonymity research for key exchange protocols exists. However, the comparison of these protocols and protocol extensions is out of scope for this work.

4.3 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.

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 and HIP, 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 4.1).

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[1].

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

[1] Note that other secure broadcast approaches based on public-key cryptography or hash chains might be considered, but they need to be analyzed with regard to specific resource-limitations.

the cryptographic overhead compared to application-focused security solutions (TLS/DTLS). 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 [28, 29]. 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.

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 [30][31] 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 [32], 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 [33]. However, slight adjustments might be necessary to reduce the cryptographic costs, for example, by making the public-key signatures in the mobility messages optional[2]. 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 [34].

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 net-

[2]The signature serves for the purpose of supporting HIP-aware middleboxes in verifying the authenticity of HIP signaling messages and can be removed if end-to-middle authentication is not needed. The HMAC included in each mobility update message will still allow for end-to-end authentication and integrity protection.

works. However, if individual *things* change their point of network attachment while communicating, mobility support may gain importance.

5 Conclusions

Starting from the lifecycle of a *thing* in a BAC application, this paper reviewed the architectural design for a secure IP-based Internet of Things and its challenges with special focus on standard IP security protocols.

A first conclusion refers to the fact that the security architecture should fit the lifecycle of a *thing* and its capabilities. This includes aspects such as the way a security domain is created, the need for a trusted-third party in this process, or the type of protocols applied. Another important requirement for an architecture is fact that it should scale from small-scale ad-hoc security domains of *things* to large-scale deployments, potentially spanning several security domains. Security protocols should further take into account the resource-constrained nature of *things* and heterogeneous communication models. As for the first aspect, security protocols should include lightweight security mechanisms that are feasible to be run on *small things*. In order to enable end-to-end security *and* domain-specific protocol variants, protocols should be adapted to support translations done by gateways. Group security must be considered as well, since the IoT brings communication patterns that are unusual in traditional networks, and thus are not sufficiently supported by end-to-end Internet security protocols. Protocol design should further take into account the effect of packet fragmentation on security, with particular focus on possible DoS attacks.

Beyond these challenges, the question, at which level to base the security in the IoT, is of great importance. The link layer, the network layer, as well as the application layer have distinct security requirements and communication patterns. For small devices, resource limitations make it challenging to secure all layers individually. Securing only the application layer leaves the network open to attacks, while security focused only at the network and link layer might introduce possible inter-application security threats. Hence, the limited resources of *things* may require sharing of keying material and common security mechanisms between layers. Such cross layer concepts should be considered for an IoT-driven redesign of Internet security protocols. As future work, we aim at a deeper feasibility analysis of the discussed protocols in different settings and for different trust models.

References

1. AUTO-ID LABS. http://www.autoidlabs.org/. online, last visited 30. June 2011.
2. E. Kim, D. Kaspar, N. Chevrollier, and JP. Vasseur. Design and Application Spaces for 6LoWPANs draft-ietf-6lowpan-usecases-09. Design and Application Spaces for 6LoWPANs draft-ietf-6lowpan-usecases-09, January 2011.
3. BACnet. http://www.bacnet.org/. online, last visited 30. June 2011.
4. DALI. http://www.dalibydesign.us/dali.html. online, last visited 25 Feb. 2011.
5. ZigBee. http://www.zigbee.org/. online, last visited 30. June 2011.

6. IETF 6LoWPAN Working Group. http://tools.ietf.org/wg/6lowpan/. online, last visited 30. June 2011.
7. G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, September 2007.
8. IETF Constrained RESTful Environment (CoRE) Working Group. https://datatracker.ietf.org/wg/core/charter/. online, last visited 30. June 2011.
9. Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP). draft-ietf-core-coap-04 (Internet Draft), January 2011.
10. C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306, December 2005. Updated by RFC 5282.
11. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008. Updated by RFCs 5746, 5878.
12. T. Phelan. Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP). RFC 5238, May 2008.
13. R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201 (Experimental), April 2008.
14. R. Moskowitz, P. Jokela, T. Henderson, and T. Heer. Host Identity Protocol Version 2. draft-ietf-hip-rfc5201-bis-03 (Work in progress), October 2011.
15. D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. Protocol for Carrying Authentication for Network Access (PANA). RFC 5191, May 2008.
16. B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748, June 2004.
17. T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, January 2006.
18. R. Moskowitz. HIP Diet EXchange (DEX). draft-moskowitz-hip-rg-dex-05 (Work in progress), 2011.
19. H. Chan, A. Perrig, and D Song. Random key predistribution schemes for sensor networks. In *in Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003.
20. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D Tygar. Spins: Security protocols for sensor networks. In *in Wireless Networks Journal, September 2002*, 2002.
21. M Langheinrich. *Personal Privacy in Ubiquitous Computing*. PhD thesis, ETH Zurich, 2005.
22. V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded internet. In *in Proceedings of PerCom 2005*, 2005.
23. S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). RFC 4492 (Informational), May 2006. Updated by RFC 5246.
24. D. Fu and J. Solinas. Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2. RFC 5903 (Informational), June 2010.
25. M. Dworkin. NIST Special Publication 800-38B. *NIST Special Publication*, 800(38B):38B, 2005.
26. B. Sarikaya, Y. Ohba, Z. Cao, and R. Cragie. Security Bootstrapping of Resource-Constrained Devices. Security Bootstrapping of Resource-Constrained Devices, January 2011.
27. P. Duffy, S. Chakrabarti, R. Cragie, Y. Ohba, and A. Yegin. Protocol for Carrying Authentication for Network Access (PANA) Relay Element. draft-ohba-pana-relay-03 (Work in progress), February 2011.

28. MSEC WG website. http://datatracker.ietf.org/wg/msec/. online, last visited 30. June 2011.
29. J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing. RFC 3830, August 2004. Updated by RFC 4738.
30. P. Eronen. IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC 4555, June 2006.
31. T. Kivinen and H. Tschofenig. Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol. RFC 4621 (Informational), August 2006.
32. P. Nikander and J. Melen. A Bound End-to-End Tunnel (BEET) mode for ESP. draft-nikander-esp-beet-mode-09 (Work in progress), February 2009.
33. P. Nikander, T. Henderson, C. Vogt, and J. Arkko. End-Host Mobility and Multihoming with the Host Identity Protocol. RFC 5206 (Experimental), April 2008.
34. M. Williams and J. Barrett. Mobile DTLS. draft-barrett-mobile-dtls-00 (Work in progress), September 2009.