CoRE
Internet-Draft
Intended status: Informational
Expires: March 15, 2014

O. Garcia-Morchon
S. Kumar
Philips Research
S. Keoh
University of Glasgow
R. Hummen
RWTH Aachen
R. Struik
Struik Consultancy
September 11, 2013

Security Considerations in the IP-based Internet of Things
draft-garcia-core-security-06

Abstract

A direct interpretation of the Internet of Things concept refers to the usage of standard Internet protocols to allow for human-to-thing or thing-to-thing communication. Although the security needs are well-recognized, it is still not fully clear how existing IP-based security protocols can be applied to this new setting. This Internet-Draft first provides an overview of security architecture, its deployment model and general security needs in the context of the lifecycle of a thing. Then, it presents challenges and requirements for the successful roll-out of new applications and usage of standard IP-based security protocols when applied to get a functional Internet of Things.

Status of this Memo

Copyright Notice

Table of Contents

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 next steps 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 aspects within the IoT domain. In Section 4, we review existing protocols and work done in the area of security for wireless sensor networks. Section 5 identifies general challenges and needs for an IoT security protocol design and discusses existing protocols and protocol proposals against the identified requirements. Section 6 proposes a number of illustrative security suites describing how different applications involve distinct security needs. Section 7 includes final remarks and conclusions.

3. The Thing Lifecycle and Architectural Considerations

We consider the installation of a Building Automation and Control (BAC) system to illustrate the lifecycle of a thing in a BAC scenario. 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 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 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 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. 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.

At present, BAC systems use legacy building control standards such as BACNet [BACNET] or DALI [DALI] 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.

```
            _Manufactured           _SW update          _Decommissioned
   /                       /                   /
   |   _Installed          |   _ Application   |   _Removed &
   |  /                    |  / reconfigured   |  /  replaced
   |  |   _Commissioned    |  |                |  |
   |  |  /                 |  |                |  |   _Reownership &
   |  |  |    _Application |  |   _Application |  |  / recommissioned
   |  |  |   /   running   |  |  / running     |  |  |
   |  |  |   |             |  |  |             |  |  |             \\
   +##+##+###+#############+##+##+#############+##+##+##############>>>
       \/  \______________/ \/  \_____________/ \___/         time //
       /           /         \          \           \
   Bootstrapping   /     Maintenance &    \    Maintenance &
                  /     re-bootstrapping   \  re-bootstrapping
           Operational                Operational
```

The lifecycle of a thing in the Internet of Things.

Figure 1

3.1. Threat Analysis

This section explores the security threats and vulnerabilities of a network of things in the IoTs. Security threats have been analyzed in related IP protocols including HTTPS [RFC2818], 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, with regard to different phases in the thing's lifecycle. Note that these set of threats might go beyond the scope of Internet protocols but we gather them here for the sake of completeness.

1 Cloning of things: During the manufacturing process of a thing, an untrusted manufacturer can easily clone the physical characteristics, firmware/software, or security configuration of the thing. Subsequently, such a cloned thing may be sold at a cheaper price in the market, and yet be still 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. One should note here, that an untrusted manufacturer may also change functionality of the cloned thing, resulting in degraded functionality with respect to the genuine thing (thereby, inflicting potential reputational risk to the original thing manufacturer). Moreover, it can implement additional functionality with 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 reputational damage on 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. 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: The commissioning phase may also be vulnerable to man-in-the-middle attacks, e.g., when keying material between communicating entities is 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 on or sit in between the two communicating entities during the execution of this protocol. Additionally, device authentication or device authorization may be nontrivial, or may need support of a human decision process, since things usually do not have a priori knowledge about each other and can, therefore, not 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 with malicious software, thereby influencing the operational behaviour 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.

6 Extraction of security parameters: A thing deployed in the ambient environment (such as sensors, actuators, etc.) is usually physically unprotected and could easily be captured by an attacker. Such an attacker may then attempt to extract security information such as keys (e.g., device's key, private-key, group key) from this thing or try and re-program it to serve his needs. If a group key is used and compromised this way, the whole network may be compromised as well. Compromise of a thing's unique key has less security impact, since only the communication channels of this particular thing in question are compromised. Here, one should caution that compromise of the communication channel may also compromise all data communicated over this channel. In particular, one has to be weary of, e.g., compromise of group keys communicated over this channel (thus, leading to transitive exposure ripple effects).

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 anything to 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 behaviour 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 behavioural patterns of the user of interest to him. Such information can subsequently be sold to interested parties for marketing purposes and targeted advertizing.

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 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.

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

	Manufacturing	Installation/ Commissioning	Operation
Thing's Model	Device Cloning	Substitution	Privacy threat Extraction of security params
Application Layer		RFC2818 RFC4016	RFC2818, Firmware replacement
Transport Layer		Eavesdropping & Man-in-the-middle attack RFC4919, RFC5713 RFC3833, RFC3756	Eavesdropping Man-in-the-middle
Network Layer			RFC4919,DoS attack Routing attack RFC3833
Physical Layer			DoS attack

The security threat analysis

Figure 2

3.2. 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, 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 solid 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 context of the IoT, however, the security must not only focus on the required security services, but also 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:

1   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.

2   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.

3   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 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 networked things. Network security can include a number of mechanisms ranging from secure routing to data link layer and network layer security.

5   Application security guarantees that only trusted instances of an application running in the IoT can communicate with each other, while illegitimate instances cannot interfere.

```
              ..........................
              :           +-----------+:
              :       *+*>|Application|*****
              :       *|  +-----------+:   *
              :       *|  +-----------+:   *
              :       *|->| Transport |:   *
              :    * _*|  +-----------+:   *
              :    *|  |  +-----------+:   *
              :    *|  |->|  Network  |:   *
              :    *|  |  +-----------+:   *
              :    *|  |  +-----------+:   *    *** Bootstrapping
              :    *|  +->|     L2    |:   *    ~~~ Application Security
              :    *|     +-----------+:   *
              :+--------+              :   *
              :|Security| Configuration:   *
              :|Service |   Entity     :   *
              :+--------+              :   *
              :........................:   *
                                           *
  .........................                *  .........................
  :+--------+             :                *  :             +--------+:
  :|Security|   Node B    :                *  :   Node A    |Security|:
  :|Service |             :                *  :             |Service |:
  :+--------+             :                *  :             +--------+:
  :    |     +-----------+:                *  :+-----------+     |*   :
  :    |  +->|Application|:                ****|Application|<*+* |*   :
  :    |  |  +-----------+:                   :+-----------+  |* |*   :
  :    |  |  +-----------+:                   :+-----------+  |* |*   :
  :    |  |->| Transport |~~~~~~~~~~~~~~~~~~~~~| Transport |<-|* |*   :
  :    |__|  +-----------+: ................. :+-----------+  |*_|*   :
  :       |  +-----------+: : +-----------+ : :+-----------+  | *     :
  :       |->|  Network  |: : |  Network  | : :|  Network  |<-|       :
  :       |  +-----------+: : +-----------+ : :+-----------+  |       :
  :       |  +-----------+: : +-----------+ : :+-----------+  |       :
  :       +->|     L2    |: : |     L2    | : :|     L2    |<-+       :
  :          +-----------+: : +-----------+ : :+-----------+          :
  :.......................: :...............: :.......................:
```

Overview of Security Mechanisms.

Figure 3

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). Inspired by the security framework for routing over low power and lossy network [ID-Tsao], we show an example of security model and 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 for secure communication. Things can then securely communicate with each other during their operational phase by means of the employed network and application security mechanisms.

4. State of the Art

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) [ID-CoAP], that runs over UDP and enables efficient application-level communication for things.

4.1. 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 discuss a number of protocols and procedures that have been recently discussed in the context of the above working groups. The considered protocols are IKEv2/IPsec [RFC4306], TLS/SSL [RFC5246], DTLS [RFC5238], HIP [RFC5201][ID-Moskowitz], PANA [RFC5191], and EAP [RFC3748] in this Internet-Draft. Application layer solutions such as SSH [RFC4251] also exist, however, these are currently not considered. Figure 3 depicts the relationships between the discussed protocols in the context of the security terminology introduced in Section 3.1.

```
             ..........................
             :           +-----------+:
             :       *+*>|Application|*****     *** Bootstrapping
             :       *|  +-----------+:   *     ### Application Security
             :       *|  +-----------+:   *     === Network security
             :       *|->| Transport |:   *
             :    * _*|  +-----------+:   *
             :    *|  |  +-----------+:   *
             :    *|  |->|  Network  |:   *--> -PANA/EAP
             :    *|  |  +-----------+:   *    -HIP
             :    *|  |  +-----------+:   *
             :    *|  +->|     L2    |:   *     ## DTLS
             :    *|     +-----------+:   *     ##
             :+--------+              :   *
             :|Security| Configuration:   *     [] HIP,IKEv2
             :|Service |   Entity     :   *     [] ESP/AH
             :+--------+              :   *
             :........................:   *
                                          *
 .........................                *    .........................
 :+--------+             :                *    :             +--------+:
 :|Security|   Node B    :                *    :   Node A    |Security|:
 :|Service |             :                *    :             |Service |:
 :+--------+             :     Secure     *    :             +--------+:
 :    |     +-----------+:     routing    *    :+-----------+     |*   :
 :    |  +->|Application|:    framework   ******|Application|<*+* |*   :
 :    |  |  +----##-----+:        |            :+----##-----+  |* |*   :
 :    |  |  +----##-----+:        |            :+----##-----+  |* |*   :
 :    |  |->| Transport |#########|#############| Transport |<-|* |*   :
 :    |__|  +----[]-----+:  ......|..........  :+----[]-----+  |*_|*   :
 :       |  +====[]=====+=====+===========+=====+====[]=====+  | *     :
 :       |->|| Network  |:  : |  Network  | :  :|  Network ||<-|       :
 :       |  +|----------+:  : +-----------+ :  :+----------|+  |       :
 :       |  +|----------+:  : +-----------+ :  :+----------|+  |       :
 :       +->||    L2    |:  : |     L2    | :  :|     L2   ||<-+       :
 :          +===========+=====+===========+=====+===========+          :
 :.......................:  :...............:  :.......................:
```

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.

4.2. 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 [PROC-Chan] or more centralized solutions, such as SPINS [JOURNAL-Perrig], have been proposed for key establishment in wireless sensor networks. The ZigBee standard [ZB] 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 [THESIS-Langheinrich]. 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. [PROC-Gupta], 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.

5. 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 1 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.

5.1. 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.

5.1.1. 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.

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

Relationships between IP-based security protocols.

Figure 5

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 cryptoprimitives such as public-key cryptography as used in most Internet security standards. This is especially true, if the basic cryptoblocks 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][ID-Moskowitz][ID-HIP]. 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 [NIST] 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 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, judgments on the energy consumption of a particular protocol cannot be made without tailor-made IoT implementations.

5.1.2. 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 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.

5.1.3. 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:

1   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.

2   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.

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). 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, they usually offer an end-to-end secured connection. An exception is the usage layered approach as might be PANA and EAP. In such a case, this configuration (i) allows 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.

5.2. 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. 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.

5.2.1. 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. 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.

5.2.2. 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. It is important to realize that bootstrapping may involve different types of information, ranging from network parameters and information on device capabilities and their presumed functionality, to management information related to, e.g., resource scheduling and trust initialization/management. Furthermore, bootstrapping may occur in stages during the lifecycle of a device and may include provisioning steps already conducted during device manufacturing (e.g., imprinting a unique identifier or a root certificate into a device during chip testing), further steps during module manufacturing (e.g., setting of application-based configurations, such as temperature read-out frequencies and push-thresholds), during personalization (e.g., fine-tuned settings depending on installation context), during hand-over (e.g., transfer of ownership from supplier to user), and, e.g., in preparation of operation in a specific network. In what follows, we focus on bootstrapping of security-related information, since bootstrapping of all other information can be conducted as ordinary secured communications, once a secure and authentic channel between devices has been put in place.

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. 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 host 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 host.

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 [ID-OFlynn] 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 other 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 [RFC6345], the PANA relay element is introduced, relaying PANA messages between a PaC (joining thing) and PAA of a segment [ID-OFlynn]. This approach forces commissioning based on distance to PAA, i.e., things can only be bootstrapped hop-by-hop starting from those closer to the PAA, all things that are 1-hop away are bootstrapped first, followed by those that are 2-hop away, and so on. Such an approach might impose important limitations on actual use cases in which, e.g., an installer without technical background has to roll-out the system.

5.2.3. 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 here discussed protocols, 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.

5.3. 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.

5.3.1. 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 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 5.1).

5.3.2. 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). 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.

5.3.3. 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.

6. Security Suites for the IP-based Internet of Things

Different applications have different security requirements and needs and, depending on various factors, such as device capability, availability of network infrastructure, security services needed, usage, etc., the required security protection may vary from "no security" to "full-blown security". For example, applications may have different needs regarding authentication and confidentiality. While some application might not require any authentication at all, others might require strong end-to-end authentication. 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 6LoWPAN network to distribute and manage keys; while other applications cannot rely on such a central party or their availability.

Thus, it is essential to define security profiles to better tailor security solutions for different applications with the same characteristics and requirements. This provides a means of grouping applications into profiles and then defines the minimal required security primitives to enable and support the security needs of the profile. The security elements in a security profile can be classified according to Section 3.1, namely:

1   Security architecture,

2   Security model,

3   Security bootstrapping,

4   Network security, and

5 Application security.

In order to (i) guide the design process by identifying open gaps; (ii) allow for later interoperability; and (iii) prevent possible security misconfigurations, this section defines a number of generic security profiles with different security needs. Each security profile is identified by:

1 a short description,

2 an exemplary application that might use/require such a security policy,

3 the security requirements for each of the above security aspects according to our classification in Section 3.1.

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. It 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 misconfigurations, since each security profile can be bound to a different application area so that it can be clearly defined which security protocols and approaches can be applied where and under which circumstances.

Note that each of these security profiles aim at summarizing the required security requirements for different applications 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 of the Internet Draft, 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. These security profiles are summarized in the table below:

	Application	Description
SecProf_0	No security needs	6LoWPAN/CoAP is used without security
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

Security profiles and application areas.

Figure 6

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

SecProf_0 is meant for any application that does not require security. Examples include applications during system development, system testing, or some very basic applications in which security is not required at all.

The second security suite (SecProf_1) is catered for environments in which 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 that it usually happen 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 interoperate 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 locally. 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 6LoWPAN/CoAP network can be prone to Denial of Service attacks (e.g., flooding if L2 is not protected) or routing attacks.

SecProf_3 considers that a central device is always required for network management. Example applications of this profile include building control and automation, sensor networks for industrial use, environmental monitoring, etc. As before, the network manager can be located in the 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. In other words, the threat model requires measurements to protect against any vulterable period of time. 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 [ID-CoAPMulticast]. Network should be protected from attacks. A further extension to this use case is to allow for remote management. A "backend manager" is in charge of managing SW or information exchanged or collected within the 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-proHTTPCoAP], etc. This use case requires protecting the communication from a device in the backend to a device in the 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 6LoWPAN/CoAP security domains. Such operational requirements pose additional security requirements, e.g., in addition to secure bootstrapping of a device within a 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.

The above description illustrates how different applications of 6LoWPAN/CoAP networks involve different security needs. In the following sections, we 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. Security Architecture

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

	Description
SecProf_0	-
SecProf_1	Distributed
SecProf_2	Distributed able to move centralized (local)
SecProf_3	Centralized (local &/or backend)
SecProf_4	Distributed & centralized (local &/or backend)

Security architectures in different security profiles.

Figure 7

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 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 keying materials from the backend.

6.2. 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 advance 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.

	Description
SecProf_0	-
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
SecProf_4	(no) Tamper resistant Sharing keys between layers/Key and process separation Sandbox

Thing security models in different security profiles.

Figure 8

6.3. 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]. 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 boostrapping 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_0	-
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.

Security boostrapping methods in different security profiles

Figure 9

6.4. Network Security

Network security refers to the mechanisms used to ensure the secure transport of 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_0	-
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.

Network security needs in different security profiles

Figure 10

6.5. Application Security

In the context of 6LoWPAN/CoAP networks, application security refers firstly to the configuration of DTLS used 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_0	-
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

Application security methods in different security profiles

Figure 11

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. Furthermore, it can involve the usage of group keys for group communication. If secure multicast is implemented, it should provide origin authentication. 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. Next Steps towards a Flexible and Secure Internet of Things

This Internet Draft included an overview of both operational and security requirements of things in the Internet of Things, discussed a general threat model and security issues, and introduced a number of potential security suites fitting different types of IoT deployments.

We conclude this document by giving our assessment of the current status of CoAP security with respect to addressing the IP security challenges we identified, so as to facilitate discussion of next steps towards workable security design concepts suitable for IP-based IoT in the broader community. Hereby, we focus on the employed security protocols and the type of security architecture.

With current status, we refer to the feasibility of realizing secure deployments with existing CoAP protocols and the practicality of creating comprehensive security architectures based on those protocols:

1   DTLS has been defined as the basic building block for protecting CoAP. At the time it was first proposed, no DTLS implementation for small, constrained devices was available. In the mean-time, TinyDTLS [TinyDTLS] has been developed offering the first open-source implementation of the protocol for small devices. However, more experience with the protocol is required. In particular, a performance evaluation and comparison should be made with a well-defined set of standard node platforms/networks. The results will help understand the limitations and the benefits of DTLS as well as to give recommended usage scenarios for this security protocol.

2   (D)TLS was designed for traditional computer networks and, thus, some of its features may not be optimal for resource-constrained networks. This includes:

    a   Basic DTLS features that are, in our view, not ideal for resource-constrained devices. For instance, the loss of a message in-flight requires the retransmission of all messages in-flight. On the other hand, if all messages in-flight are transmitted together in a single UDP packet, more resources are required for handling of large buffers. As pointed out in [ID-Hartke] , the number of flights in the DTLS handshake should be reduced, so that a faster setup of a secure channel can be realized. This would definitely improve the performance of DTLS significantly.

    b   Fragmentation of messages due to smaller MTUs in resource-constrained networks is problematic. This implies that the node must have a large buffer to store all the fragments and subsequently perform re-ordering and reassembly in order to construct the entire DTLS message. The fragmentation of the handshake messages can, e.g., allow for a very simple method to carry out a denial of service attack.

    c   The completion of the DTLS handshake is based on the successful verification of the Finished message by both client and server. As the Finished message is computed based on the hash of all handshake messages in the correct order, the node must allocate a large buffer to queue all handshake messages.

    d   DTLS is thought to offer end-to-end security; however, end-to-end security also has to be considered from the point of view of LLN protection, so that end-to-end exchanges can still be verified and the LLN protected from, e.g., DoS attacks.

3   Raw public-key in DTLS has been defined as mandatory. However, memory-optimized public-key libraries still require several KB of flash and several hundreds of B of RAM. Although Moore's law still applies and an increase of platform resources is expected, many IoT scenarios are cost-driven, and in many use cases, the same work could be done with symmetric-keys. Thus, a key question is whether the choice for raw public-key is the best one. In addition, using raw public keys rather than certified public keys hard codes identities to public keys, thereby inhibiting public key updates and potentially complicating initial configuration.

4   Performance of DTLS from a system perspective should be evaluated involving not just the cryptographic constructs and protocols, but should also include implementation benchmarks for security policies, since these may impact overall system performance and network traffic (an example of this would be policies on the frequency of key updates, which would necessitate securely propagating these to all devices in the network).

5   Protection of lower protocol layers is a must in networks of any size to guarantee resistance against routing attacks such as flooding or wormhole attacks. The wireless medium that is used by things to communicate is broadcast in nature and allows anybody on the right frequency to overhear and even inject packets at will. Hence, IP-only security solutions may not suffice in many IoT scenarios. At the time of writing the document, comprehensive methods are either not in place or have not been evaluated yet. This limits the deployment of large-scale systems and makes the secure deployment of large scale networks rather infeasible.

6   The term "bootstrapping" has been discussed in many occasions. Although everyone agrees on its importance, finding a good solution applicable to most use cases is rather challenging. While usage of existing methods for network access might partially address bootstrapping in the short-term and facilitate integration with legacy back-end systems, we feel that, in the medium-term, this may lead to too large of an overhead and imposes unnecessary constraints on flexible deployment models. The bootstrapping protocol should be reusable and light-weight to fit with small devices. Such a standard bootstrapping protocol must allow for commissioning of devices from different manufacturers in both centralized and ad-hoc scenarios and facilitate transitions of control amongst devices during the device's and system's lifecycle. Examples of the latter include scenarios that involve hand-over of control, e.g., from a configuration device to an operational management console and involving replacement of such a control device. A key challenge for secure bootstrapping of a device in a centralized architecture is that it is currently not feasible to commission a device when the adjacent devices have not been commissioned yet. In view of the authors, a light-weight approach is still required that allows for the bootstrapping of symmetric-keys and of identities in a certified public-key setting.

7   Secure resource discovery has not been discussed so far. However, this issue is currently gaining relevance. The IoT, comprising sensors and actuators, will provide access to many resources to sense and modify the environment. The usage of DNS presents

well-known security issues, while the application of secure DNS may not be feasible on small devices. In general, security issues and solutions related to resource discovery are still unclear.

8 A security architecture involves, beyond the basic protocols, many different aspects such as key management and the management of evolving security responsibilities of entities during the lifecycle of a thing. This document discussed a number of security suites and argued that different types of security architectures are required. A flexible IoT security architecture should incorporate the properties of a fully centralized architecture as well as allow devices to be paired together initially without the need for a trusted third party to create ad-hoc security domains comprising a number of nodes. These ad-hoc security domains could then be added later to the Internet via a single, central node or via a collection of nodes (thus, facilitating implementation of a centralized or distributed architecture, respectively). The architecture should also facilitate scenarios, where an operational network may be partitioned or merged, and where hand-over of control functionality of a single device or even of a complete subnetwork may occur over time (if only to facilitate smooth device repair/ replacement without the need for a hard "system reboot" or to realize ownership transfer). This would allow the IoT to transparently and effortlessly move from an ad-hoc security domain to a centrally-managed single security domain or a heterogeneous collection of security domains, and vice-versa. However, currently, these features still lack validation in real-life, large-scale deployments.

9 Currently, security solutions are layered, in the sense that each layer takes care of its own security needs. This approach fits well with traditional computer networks, but it has some limitations when resource-constrained devices are involved and these devices communicate with more powerful devices in the back-end. We argue that protocols should be more interconnected across layers to ensure efficiency as resource limitations make it challenging to secure (and manage) all layers individually. In this regard, securing only the application layer leaves the network open to attacks, while security focused only at the network or 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. It is required that the data format of the keying material is standardized to facilitate cross-layer interaction. Additionally, cross-layer concepts should be considered for an IoT-driven re-design of Internet security

protocols.

8. Security Considerations

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

9. IANA Considerations

This document contains no request to IANA.

10. Acknowledgements

We gratefully acknowledge feedback and fruitful discussion with Tobias Heer and Robert Moskowitz.

11. References

11.1. Informative References

[RFC6568]Kim, E., Kaspar, D., and JP. Vasseur, "Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6568, April 2012.

[RFC2818]Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[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, August 2011.

[ID-CoAP]Z. Shelby, K. Hartke, C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18, June 2013.

[ID-CoAPMulticast]Rahman, A. and E. Dijk, "Group Communication for CoAP",draft-ietf-core-groupcomm-12 (work in progress), July 2013.

[ID-Daniel]Park, S., Kim, K., Haddad, W., Chakrabarti, S., and J. Laganier, "IPv6 over Low Power WPAN Security Analysis",Internet Draft draft-daniel-6lowpan-security-analysis-05, Mar 2011.

[ID-HIP]Moskowitz, R., "HIP Diet EXchange (DEX)", draft-moskowitz-hip-rg-dex-06 (work in progress), May 2012.

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

[ID-Moskowitz]Moskowitz, R., Heer, T., Jokela, P., and Henderson, T., "Host Identity Protocol Version 2", draft-ietf-hip-rfc5201-bis-13 (work in progress), Sep 2013.

[ID-Nikander]Nikander, P. and J. Melen, "A Bound End-to-End Tunnel(BEET) mode for ESP", draft-nikander-esp-beet-mode-09, Aug 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 (work in progress), Nov 2010.

[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, Jan 2012.

[ID-Williams]Williams, M. and J. Barrett, "Mobile DTLS", draft-barrett-mobile-dtls-00, Mar 2009.

[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(work in progress), Feb 2013.

[RFC3261]Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3748]Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3756]Nikander, P., Ed., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[RFC3833]Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, August 2004.

[RFC4016]Parthasarathy, M., "Protocol for Carrying Authentication and Network Access (PANA) Threat Analysis and Security Requirements", RFC 4016, March 2005.

[RFC5246]Dierks, T. and E. Rescorla, "The Transport Layer Security

(TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC4251]Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4306]Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4555]Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.

[RFC4621]Kivinen, T. and H. Tschofenig, "Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol", RFC 4621, August 2006.

[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, November 2006.

[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, August 2007.

[RFC4944]Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC5191]Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.

[RFC5201]Moskowitz, R., Nikander, P., Jokela, P., Ed., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

[RFC5206]Nikander, P., Henderson, T., Ed., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.

[RFC5238]Phelan, T., "Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP)", RFC 5238, May 2008.

[RFC5246]Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5713]Moustafa, H., Tschofenig, H., and S. De Cnodder, "Security Threats and Security Requirements for the Access Node Control Protocol (ANCP)", RFC 5713, January 2010.

[RFC5903]Fu, D. and J. Solinas, "Elliptic Curve Groups modulo a Prime

(ECP Groups) for IKE and IKEv2", RFC 5903, June 2010.

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

[BACNET]"BACnet", Web http://www.bacnet.org/, Feb 2011.

[DALI]"DALI", Web http://www.dalibydesign.us/dali.html, Feb 2011.

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

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

[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, Nov 1999.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[THESIS-Langheinrich]Langheinrich, M., "Personal Privacy in Ubiquitous Computing", PhD Thesis ETH Zurich, 2005.

[TinyDTLS "TinyDTLS", Web http://tinydtls.sourceforge.net/, Feb 2012.

[WG-6LoWPAN]"IETF 6LoWPAN Working Group", Web http://tools.ietf.org/wg/6lowpan/, Feb 2011.

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

[WG-MSEC]"MSEC Working Group", Web http://datatracker.ietf.org/wg/msec/.

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

Authors' Addresses

Oscar Garcia-Morchon
Philips Research
High Tech Campus
Eindhoven,   5656 AA
The Netherlands

Email: oscar.garcia@philips.com

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

Email: sandeep.kumar@philips.com

Sye Loong Keoh
University of Glasgow Singapore
Republic PolyTechnic, 9 Woodlands Ave 9
Singapore 838964
SG

Email: SyeLoong.Keoh@glasgow.ac.uk

Rene Hummen
RWTH Aachen University
Templergraben 55
Aachen,   52056
Germany

Email: rene.hummen@cs.rwth-aachen.de

Rene Struik
Struik Security Consultancy
Toronto,
Canada

Email: rstruik.ext@gmail.com

CoRE
Internet-Draft
Intended status: Informational
Expires: March 10, 2014

B. Greevenbosch
Huawei Technologies
September 06, 2013

Use cases and requirements for authentication and authorisation in CoAP
draft-greevenbosch-core-authreq-00

Abstract

This draft describes use cases and requirements for authenticated and authorised CoAP. The draft especially focuses on threats and their prevention.

Note

Discussion and suggestions for improvement are requested, and should be sent to core@ietf.org.

Status of This Memo

Copyright Notice

Table of Contents

1. Requirements notation

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 [RFC2119].

2. Introduction

This draft describes use cases and requirements for secure authentication and authorisation, as well as their expiry and revocation, in CoAP.

The draft consists of the following parts:

o The draft starts with several use cases.

- A section with requirements related to the use cases follows.

- Discussion of the various security trade-offs that need to be made can be found in Section 5.

The goal of the draft is to provide background material for usage when defining a solution for authorised CoAP.

3. Use cases

3.1. Authorised and unauthorised devices

Company A produces sensor devices. These devices are of high quality, and no vulnerabilities have been detected. As such, they have been certified to be used in a wide area of applications.

Company B produces also sensor devices. However, these devices are of low quality, and have known security issues. They failed the certification requirements.

Company C is oblivious of this fact, and since it needs this kind of sensors to monitor its industrial process, it buys some to test.

During installation of the sensors into Company C’s monitoring network, the credentials of the sensors are verified by the system. The sensors from Company A install without problem. However, for the sensors from Company B the authentication fails, and the installation of the sensors is refused. The system informs the installation engineers about the reason of failure.

Fortunately the authentication mechanism revealed that the sensors from Company B are not to be used. This avoided a lot of trouble and potential security issues.

3.2. Home security

Henry has an advanced home security system. The security system provides protection against burglary, as well as against fire. It has sensors on doors, motion sensors, smoke detectors, cameras etc. It also has actuators for the electronic locks, a sprinkler system and actuators that can close the gas tap and cut the electricity.

The system comes with tokens. These tokens are used to turn on or off part of the system, and allow certain actions that need human interaction. One of these actions is to open or close the front door lock. Henry has provided a token to each of his family members.

The system has a solid authorisation and authentication model, ensuring that only Henry and his family's tokens can drive the system. Even though the tokens can be bought in a regular store, only tokens that Henry has approved can be used in the system.

Certain peripherals allow different access rights to different entities. For example, the electricity closure can only be set by Henry and the master system, whereas its on/off status can be read by all family members.

All peripherals are certified by an impartial certification body, which has specified minimum security requirements. In this way, Henry is assured that when he adds a new peripheral and it is accepted by the system, it can be deemed reliable.

3.3. Illegal smart-meters

An electricity company depends on smart-meters to measure energy usage of the households it servers. The gathered information is used for several purposes, billing being one of them.

On the black market, there appear illegal smart-meters that only report 75% of the actual electricity usage. These smart-meters are based on a clone of a valid public key.

Once the electricity company discovers this, it revokes the associated public key, thereby ensuring that the illegal meters cannot be installed anymore.

3.4. Maintaining and extending a network of sensors and actuators

An agricultural company uses an IP network to ensure an optimal climate for the vegetables they grow in their green houses. Sensors do measurements about e.g. humidity and sunlight, whereas actuators can drive artificial rain and supporting light. A central controller is responsible for processing the sensor readings and driving the actuators accordingly.

Sometimes, a sensor or actuator needs replacement as part of the normal maintenance cycle. This is a routine task for the associated engineer, and involves simply disconnecting the old apparatus and connecting a new one. The rest of the installation to the network happens automatically.

As the agricultural company is doing good business, it decides to expand. It buys another piece of land, and modernises the green house that was already built on the land. The modernisation includes installing new sensors and actuators, which are seamlessly integrated into the already existent network, such that they can work with the central controller too.

The use case illustrates the need to be able to automatically install and update network nodes in an existing network. It is also important to note, that installation of the network nodes includes proper authentication and authorisation. After all, the agricultural company does not want outsiders to be able to influence the climate in the green houses, for example by driving the actuators or modifying the sensor readings.

3.5. Discovered compromised device

Company A has a certain type of actuators installed throughout its building. On a certain time, some of these actuators start behaving funny. It turns out that some hackers have been able to access the sensors, and drive them as they wish.

Company A can't de-install the actuators immediately, after all, they are installed everywhere in the building. Instead Company A has the actuators revoked, and then can replace them on a less hasty schedule.

3.6. Vulnerability discovery in actuators in a chemical plant

A chemical plant deploys sensors for the several properties of the substance being produced, and actuators that start certain processes when the substance is ready for the next step.

A vulnerability in certain of the actuators is discovered; it would allow unauthorised third parties to take over the actuators and start processes at their will.

After the discovery of the vulnerability, the chemical plant pro-actively de-activates the actuators and revokes their keys. It then makes sure the vulnerability is resolved as quickly as possible, such that normal production can resume.

3.7. Revocation of a non-compromised device

Jack worked at the IT department of company E.

However, due to a conflict with the company, Jack has been fired. When leaving, he smuggled out some tokens used to control several of the company's peripherals.

When the company realises it misses the tokens, it revokes them to ensure they cannot be used to control the peripherals anymore.

Jack fails to wreak havoc as his revenge, and neither can he sell the tokens to other adversaries.

3.8. Mixing nodes from different vendors

A weather analysis and forecast agency needs global coverage for collection of temperature and air-pressure data. It has contracts with several local authorities and companies for the placement of their sensors.

For both logistic and economic reasons, the weather agency does not want to rely on one particular type of sensor from a single vendor. Instead, it wants to allow different sensors from different vendors, as long as these sensors meet certain criteria concerning precision, response time and reliability.

To ensure the criteria are met, the weather agency performs several tests with new candidate sensors. When the sensors pass the tests, the agency allows their usage in its network. When the sensors fail the tests, the agency is ensured that they cannot be used for collecting data, lest the quality of the agency's analysis and forecast suffer from data of bad quality.

In this use case, the vendor pro-actively controls which sensor types can be used in their network. It uses an authentication and authorisation mechanism to automatically ensure that only those types it has approved can be installed. The use case illustrates the need for interoperability in authentication between nodes manifactured by different vendors, as well as the need to exclude nodes that are not authorised to join the network.

3.9. Privacy of medical communications

Mr P has developed a heart problem. To diagnose and monitor the condition of Mr P's heart, his cardiologist has requested Mr P to wear a sensor during the day. The sensor measures the heartbeat and other vital functions. The sensor transmits this information to the hospital, generally once every day. When needed, e.g. when a situation occurs that requires extra attention, the sensor can also send information ad-hoc.

Protecting the integrity of the sensor readings is important, even when it is unlikely that an adversary will tamper with the sensor readings. After all, doing so would constitute a serious crime. Protecting Mr P's privacy adds significantly to the value of a solid security model in this use case. In any case eavesdropping needs to be prevented, and that includes man-in-the-middle attacks.

4. Requirements

This section lists requirements associated with authentication and authorisation in CoAP:

1. It SHALL be possible to verify the binding between the key and the entity associated with it.

2. It SHALL be possible to verify whether an entity is authorised to establish the connection.

3. It SHALL be possible to specify authorisation for a specific resource.

4. It SHOULD be possible to specify authorisation based on the message type.

5. It SHALL NOT be possible for an unauthorised third party to establish a cryptographic relationship.

6. There SHALL be a mechanism that allows revocation of previously granted authorisation.

7. It SHALL be possible for a receiver to determine whether a key has been revoked.

8. It SHALL be possible to perform authentication, authorisation and revocation verification fully automatically.

9. The verification technology MUST NOT require much complexity on constrained entities.

10. The verification mechanism SHALL be scalable, allowing potentially millions of entities to verify authentication and authorisation.

11. It SHOULD be possible to specify an expiry date for keys and/or authorisation.

12. It SHALL be possible to revoke compromised keys.

13. Revocation SHALL NOT require physically unplugging the device.

14. There SHALL be protection against an unauthorised third party spoofing authorisation and/or revocation of keys and entities.

15. There SHOULD be protection against denial of service (DoS) attacks, as far as it is feasible.

5. Discussion

In this section, we discuss the various trade-offs that need to be made, and implications they may have.

5.1. Certificate authority

Much of a traditional Public Key Infrastructure depends on a certificate authority. The certificate authority (CA) signs the certificate of the device, or an intermediate certificate that signs the certificate of the device.

This creates islands of trust, in which the CA has the power to revoke any key on its island. Interoperability between devices of different CAs may still be possible, depending on which CAs the entities trust apart from their own CA.

5.2. Expiry

X.509 certificates [X.509] contain an expiry date. This means that the certificates automatically become invalid after a time has passed. Should the device’s lifetime be longer than the validity period of the certificate, then the certificate has to be updated.

The expiry date has the advantage that there is no need to keep track of revoked certificates infinitely. After the certificate’s expiration, the revocation status can be forgotten.

However a major draw-back is that a mechanism is needed to update expired certificates, provided that the entities holding them should continue to be used.

5.3. Time of revocation

Authentication and revocation are normally checked when two entities meet each other for the first time. But how about entities that are to be revoked later?

The dealings with this highly depends on the security requirements of the employed system. For example, home light-switches may have less stringent security requirements than actuators in a chemical plant. In the former, a revocation mechanism for deployed devices may not be needed, whereas in the latter it is essential.

6. Security considerations

This whole draft concerns security considerations. It indicates use cases and requirements for authentication, authorisation and associated expiry and revocation. In addition it discusses several of the associated details and trade-offs.

We refer to the rest of the draft for the complete picture.

7. IANA considerations

No IANA requests are required for this document.

8. Acknowledgements

Thanks to Rene Struik and Kepeng Li for their valuable feedback.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[X.509]   , "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. ", ITU-T Recommendation X.509, ISO/IEC 9594-8:2005, 2005.

Author's Address

Bert Greevenbosch
Huawei Technologies Co., Ltd.
Huawei Industrial Base
Bantian, Longgang District
Shenzhen 518129
P.R. China

Phone: +86-755-28978088
Email: bert.greevenbosch@huawei.com

CoRE Working Group
Internet-Draft
Intended Status: Informational
Expires: August 18, 2014

L. Seitz
SICS Swedish ICT
G. Selander
Ericsson

February 14, 2014

Design Considerations for Security Protocols in Constrained Environments
draft-seitz-ace-design-considerations-00

Abstract

Considerable effort has been spent on securing existing Internet standard authentication and authorization protocols such as TLS, Kerberos, and OAuth, among others. It would save a lot of effort if these protocols could be profiled to be feasible for constrained environments, with some easily obtainable security considerations.

However, these protocols were typically not designed with constrained environments in mind, so profiling of an existing protocol may result in a far from optimal solution. Moreover they are not necessarily complying with their original design objectives outside their intended domain of application.

This document examines the impact of typical characteristics of security protocols (e.g. cryptographic calculations, number and size of protocol messages) in a constrained environment. The goal is to provide decision support when different resource usage optimizations are possible in the adaptation of a security protocol for this setting.

Status of this Memo

Copyright and License Notice

Table of Contents

1. Introduction

When adapting security protocols for constrained nodes, one has to take into account the various resource limitations. While it might be tempting to optimize the usage of a certain resource (e.g. minimizing RAM consumption), such an approach might produce a less-than-optimal overall solution, compared to a more holistic approach that leverages the combined effect of different optimization possibilities.

The goal of this document is to summarize some characteristics of security protocols and weigh their impact against each other in order to allow effective trade-offs when adapting existing protocols to a constrained setting. While there is some overlap with the scope of the Lightweight Implementation Guidance WG, this document is aimed more at security protocol profiling and design than actual implementation decisions that are the main focus of LWIG.

1.1 Terminology

Certain security-related terms are to be understood in the sense defined in [RFC4949]. These terms include, but are not limited to, "authentication", "authorization", "confidentiality", "encryption", "data integrity", "message authentication code", and "verify".

Terminology for constrained environments is defined in [I-D.ietf-lwig-terminology] e.g. "constrained device".

2. Background

We are assuming a multi-party protocol setting with at least the following parties

a) a resource server hosting resources
b) a client seeking access to some resource, and
c) an authorization server acting Trusted Third Party (TTP) for key distribution and access control handling.

The resource server and/or the client is assumed to be constrained, but the authorization server is not.

The authorization server can provide authentication and authorization means (e.g. cryptographic keys, access control information, certificates) for the other parties.

There are various authentications and authorizations taking place in this multi-party protocol. For example, the client and authorization server mutually authenticates and and the client is being authorized by the authorization server. The resource server needs authenticate information provided by the authorization server, based on a previously established relationship (e.g. shared symmetric keys). Finally when the client communicates with the resource server, the client's authorization needs to be verified, which might include authentication of the client.

Note: Security protocols designed to handle authentication and authorization between two mutually unknown less-constrained peers are not necessarily adapted to the current setting, where optimizations can be made by relying on an relatively unconstrained TTP.

2.1. Device assumptions

Devices may be constrained in different ways, as described in the LWIG terminology document [I-D.ietf-lwig-terminology]. This work is targeting class 1 devices, but may be applicable even the most constrained class of devices (C0) if supported by relevant proxy functionality. Class 2 devices probably do not need any special considerations, since they can mostly support the same protocols as unconstrained devices.

A device for which these considerations apply could e.g. run the following protocol stack, potentially supported by a proxy:

- o The application layer protocol is CoAP [I-D.ietf-core-coap], using UDP at the transport layer.
- o CoAP will be running on top of DTLS [RFC6347].
- o IPv6 [RFC4291] is assumed to be the Internet layer protocol on top of the adaptation layer 6LoWPAN [RFC4944].
- o IEEE 802.15.4 [IEEE802] is assumed as the Link layer protocol for wireless communication. We assume that a large proportion of the target devices will communicate over wireless channels.

2.2 Relevant Factors

From the LWIG terminology draft [I-D.ietf-lwig-terminology] we can list the following resources that need to be considered in general:

- o RAM memory (required state and buffers for running protocols)
- o Flash/ROM memory (required libraries and code complexity)
- o Computational power (required processing speed)
- o Electrical energy (battery consumption, if not mains-powered)
- o User interface and physical accessibility (for performing manual operations directly on the device)
- o Network (bit rate, loss rate, dynamic topology, fragmentation,

lack of advanced services)

The consumers of these resources in the case of security protocols can be summarized as follows:

- Cryptographic algorithms
  - based on symmetric cryptography
  - based on asymmetric cryptography
  - (orthogonal) implemented by a co-processor (e.g. AES, SHA, ECC)
- Composing/parsing protocol messages (e.g. Base64 en/decoding, JSON, ASN.1, CBOR)
- Sending/receiving protocol messages
- Listening, while waiting to receive protocol messages

2.3 Security protocols in constrained environments

One of the potential advantages with extending basic Internet Protocols to constrained nodes is that other standardized protocols can be applied too.

In particular in the case of security protocols, there is a considerable effort spent to eliminate flaws and weaknesses that could otherwise be exploited for attacking the system. It would save a lot of effort if it was possible to profile these protocols for running efficiently in a constrained environment while maintaining their security properties.

However, the profiling of a protocol may result in a far from optimal solution. For example assume that a constrained profile of a security protocol is made by reducing the message sizes. Such a protocol may still be badly suited for constrained devices e.g. because the number of round trips is what makes the latency high, and reducing that would essentially change the security properties of the protocol.

Moreover, as many of these protocols were not designed for a constrained environment, they are not necessarily complying with their original design objectives outside their intended domain of application. Even security objectives that applied to the Internet may be violated: e.g. a DoS mitigation measure that is based on a processing commitment by a client (a "puzzle", see e.g. [RFC5201]) may be inappropriate if the server is much more constrained than the client.

This memo is intended to support the adaptation of an existing security protocol for a constrained environment by providing some considerations on resource consumption. Furthermore this memo documents the assumptions that were made as a basis for these considerations.

3. Protocol design considerations

3.1 Straightforward optimizations

This section lists some potential targets for resource optimizations.

3.1.1 Smaller messages

Reducing message size will reduce composing/parsing and sending/receiving costs which is favorably impacting energy consumption and latency. Some specific considerations:

- o Smaller than CoAP payload size (1024 bytes) avoids fragmentation at the application layer.
- o Smaller than the maximum MAC-layer frame size (e.g. 127 bytes for IEEE 802.15.4) avoids fragmentation at the link layer.
- o The largest messages are potentially those containing certificates or authorization tokens, so reducing their size significantly will have a large impact.

3.1.2 Fewer messages

Removing message exchanges or round trips have potentially large impact on energy consumption and latency.

Reducing the number of messages in a given security protocol is in general not possible without changing the essential security properties of the protocol. Experiments by Google with TLS false start [I-D.bmoeller-tls-falsestart] and TLS snap start [I-D.agl-tls-snapstart] illustrate the difficulty of trying to reduce the number of messages in an established security protocol.

Challenge-response based authentication protocols may potentially be replaced with other protocols with alternative measures to ensure freshness, such as time or sequence numbers. Such an approach would require fewer message passes, but ensuring freshness can be problematic, since some constrained devices may not be able to reliably measure time.

On the other hand, there are long lifetime battery powered IEEE 802.15.4e devices implementing Time Slotted Channel Hopping (TSCH) which has good time synchronization properties, since that is required for communication.

3.1.3 Less computations

One way of reducing the complexity of required computations is to reduce the number of public key operations used during normal operations, e.g. by keeping existing sessions alive, or generating session resumption state on a less constrained device. The drawback in this case is that either more RAM or more sending and receiving of messages are needed.

An alternative is to replace public key operations with symmetric key operations. Significant reductions in resource consumption can be achieved by using symmetric cryptography instead of asymmetric cryptography, since asymmetric cryptography generally requires larger libraries (e.g. BigInteger, elliptic curves), and consumes more RAM, processing power and energy than symmetric algorithms.

However, it is not always possible to make this replacement as some of the properties of asymmetric cryptography, such as non-repudiation of signatures, and non-confidential key distribution do not apply for symmetric keys. It may require a change in trust model, where a TTP is assumed e.g. for key management.

3.1.4 Reduce RAM usage

Reducing the usage of RAM memory can be achieved by reducing the size of variable state information required by a protocol. Different security protocols and -modes have different requirements in this respect. Optimizations may potentially be done by profiling certain options of the protocol to predefined, default values.

Another possibility is to simplify parsing and processing of protocol messages, leading to smaller libraries that need to be loaded into memory. Further the size of the protocol messages, e.g. certificates and authorization tokens, directly affects the size of the buffers that need to be allocated for receiving and sending them, so keeping them small also helps.

3.1.5 Reduce code size

The overall size of the code is influenced mainly by the size of the libraries needed for cryptography and parsing messages (ASN.1, JSON, XML). In general asymmetric cryptography requires larger libraries (e.g. BigInteger, Elliptic curves) than symmetric cryptography. Minimal libraries for parsing ASN.1 and JSON are roughly comparable in size (around 6 kB) while even minimal XML parsers generally have a significantly larger size.

3.2 Trade-offs

This section looks at the more difficult question how to weigh different optimizations against each other. We emphasize in this section the potential role of the authorization server as an enabler for some of the optimizations.

3.2.1 Fewer vs smaller messages

When comparing reduction of message size versus sending fewer messages in total, if one takes into account the overhead of setting up a bearer, it is more efficient to send longer messages than shorter messages. Considering fragmentation it is better to send messages shorter than the fragmentation limit. Therefore optimal message size seems to be just below the fragmentation limit. Note that fragmentation carries an additional performance penalty in excess of just adding the overhead of sending several fragments, since fragmenting a message increases the risk that a fragment is lost and that the message as a whole needs to be retransmitted.

3.2.2 Crypto vs message exchange

It is known that in wireless constrained devices, the energy consumption for sending and receiving messages is high, and significantly higher than symmetric crypto operations [Margi10impact] and [Meu08engery]. Hence if it is possible to send fewer messages at the cost of delegating some symmetric crypto to the constrained device, such a trade off is favorable. The potential drawback is increased latency and code size. The latter could probably be avoided by reusing existing symmetric algorithms that are needed anyway.

Results from [Meu08engery] indicate that energy consumption for public key operations is on par or greater than message exchange for a particular security protocol. However, the efficiency of processing is increasing: The processing power follows Moore’s law (up to point) and depends on the manufacturing technology while the transmission/reception power is based on laws of physics laws that don’t change with manufacturing. So processing will be more and more energy efficient (up to a point) while the transmission/reception remains almost stable in terms of energy efficiency.

3.2.3 Transmitting vs receiving messages

Results comparing energy consumption of transmitting versus receiving messages seem contradictory. While [Margi10impact] indicates that receiving a message is much cheaper in energy consumption, than sending, [Meu08engery] seems to suggest that both costs are roughly on par.

An important point from [Meu08engery] is that one should consider the cost of listening for the next message in a protocol, while the other party is performing some computations. It is not obvious how much impact smart listening techniques such as Low Power Listening (LPL) or X-MAC [Bue06xmac] have.

Our conclusion on this issue is that is warrants further investigation in order to determine whether it should influence protocol design and profiling or not.

3.2.4 Distributing costs over deployment life time

Provisioning (e.g. access control lists) has a cost which potentially may be amortized over the lifetime of a deployment. Security protocol establishment (e.g. DTLS handshake) may similarly have a high cost that but can be acceptable, if the established session can be used for a long time. The drawback is that storage or RAM memory is consumed to save the state of the provisioned data or the established protocol.

3.2.5 Outsourcing heavy computations

A method of saving computational effort is to outsource computations to a less constrained TTP e.g. authorization decisions and policy management to the authorization server. Note however that this may be changing the trust model of the original protocol, and if the constrained device needs the result of the outsourced computation, this information must be transported in a secure way which in turn incurs a non-negligible cost.

3.2.6 DoS mitigation and anti-spoofing in the Internet

As we have seen it is important in a wireless constrained environment to restrict the number of messages sent and received in a protocol.

Some Internet security protocols include DoS mitigation or anti-spoofing mechanisms such as cookies (cf. [RFC6347]) or puzzles (cf. [RFC5201]) which adds message size and/or round trips. These mechanisms were in general not designed for a constrained environment and may potentially make the protocol unnecessarily heavy without efficiently providing the desired effect.

In fact the existence of a TTP allows for more efficient mechanisms, e.g. that a client first commits or proves source address to the authorization server which can assert such properties in an authorization token verified by a constrained server.

3.2.7 Outsourcing key management

Securing communication between two mutually unknown less-constrained peers has a high cost in terms of additional round trips, e.g. to protect against requests from spoofed initiators, DoS mitigation, challenge response protocols etc. In addition, both parties are often contributing to the generation of key material, which requires exchange of data used in key generation. These costs are a consequence of the trust model and is clearly not adapted to the current setting, where optimizations can be made by relying on an relatively unconstrained authorization server.

In addition to providing authorization decisions, the authorization server may support authentication and authorization between resource server and client by e.g.

- o providing symmetric keys to support authentication (cf. Kerberos).
- o providing protected assertions containing statements about client and server, including public key certificates.

3.2.8 Verifying authorization

As noted above, it is desirable to verify authorization of a request as early as possible in a protocol, to reduce unnecessary message exchanges and processing. However, if that involves verifying a digital signature, then the operation is in itself heavily resource consuming and would preferably only take place after it is known that the request is authorized. This is obviously a "catch 22" and there are various options to attempt to design around this.

In the present case, where we assume a TTP with a previously established relationship - say a shared symmetric key - with the resource server, the legitimacy of the request may e.g. be indicated with a Message Authentication Code instead of a digital signature over an authorization decision.

Authentication of client and server may still require verification of digital signature if public keys are used. However, as noted above, the authorization server may also support key distribution and provide symmetric keys for authentication (cf. Kerberos).

4. Security Considerations

This memo deals with design considerations for security protocols, including security trade-offs that can be made to save resources, some of which will come at the cost of weakening security.

Since a security protocol itself consume resources, one factor that needs to be taken into consideration is the possibility for attackers to use these very security protocols in order to mount a denial of service attack.

Each profiled or modified security protocol must bear its own security considerations. Protocol designers need to carefully evaluate the feasibility of stronger (and thus more resource consuming) security against the risks incurred by a weaker security that is more easy to implement or execute on a constrained node.

5. IANA Considerations

This document has no actions for IANA.

6. Acknowledgements

The authors would like to thank Sumit Shingal and Vlasios Tsiatsis for contributing to the discussion and giving helpful comments.

7. References

7.1 Normative References

[I-D.ietf-core-coap]
Shelby, Z., Hartke, K., Bormann, C., and B. Frank, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, August 2007.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

7.2 Informative References

[I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-07 (work in progress), February 2014.

[IEEE802]
IEEE Computer Society, "IEEE Standard for Local and metropolitan area networks Part 15.4: Low-Rate Wireless Personal

[I-D.bmoeller-tls-falsestart]
Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", draft-bmoeller-tls-falsestart-00 (expired draft), June 2010.

[I-D.agl-tls-snapstart]
Langley, A., "Transport Layer Security (TLS) Snap Start", draft-agl-tls-snapstart-00 (expired draft), June 2010.

[Meu08engery]
Meulenaer, G., Gosset ,F., Standaert, F., and L. Vandendorpe, "On the Energy Cost of Communication and

Cryptography in Wireless Sensor Networks", proceedings of the IEEE International Conference on Wireless and Mobile Computing, 2008.

[Margi10impact]
Margi, C., Oliveira, B., Sousa, G., Simplicio, M., Barreto, P., Carvalho, T., Naeslund, M., and R. Gold, "Impact of Operating Systmes on Wireless Sensor Networks (Security) Applications and Testbeds", proceedings of the 19th International Conference on Computer Communications and Networks (ICCCN), 2010.

[Bue06xmac]
Buettner, M., Yee, G., Anderson, E., and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks", proceedings of SenSys'06, 2006

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., Ed., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

Authors' Addresses

Ludwig Seitz
SICS Swedish ICT AB
Scheelevagen 17
22370 Lund
SWEDEN
EMail: ludwig@sics.se

Goeran Selander
Ericsson
Farogatan 6
16480 Kista
SWEDEN
EMail: goran.selander@ericsson.com

ACE Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

L. Seitz, Ed.
SICS Swedish ICT AB
S. Gerdes, Ed.
Universitaet Bremen TZI
G. Selander
Ericsson
M. Mani
Itron
S. Kumar
Philips Research

October 27, 2014

ACE use cases
draft-seitz-ace-usecases-02

Abstract

Constrained devices are nodes with limited processing power, storage space and transmission capacities. These devices in many cases do not provide user interfaces and are often intended to interact without human intervention.

This document comprises a collection of representative use cases for the application of authentication and authorization in constrained environments. These use cases aim at identifying authorization problems that arise during the lifecylce of a constrained device and are intended to provide a guideline for developing a comprehensive authentication and access control solution for this class of scenarios.

Where specific details are relevant, it is assumed that the devices use the Constrained Application Protocol (CoAP) as communication protocol, however most conclusions apply generally.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Constrained devices [RFC7228] are nodes with limited processing power, storage space and transmission capacities. These devices are often battery-powered and in many cases do not provide user interfaces.

Constrained devices benefit from being interconnected using Internet protocols. However, due to the devices' limitations, commonly used security protocols are not always easily applicable. As the devices are expected to be integrated in all aspects of everyday life, the application of adequate security mechanisms is required to prevent attackers from gaining control over data or functions important to our lives.

This document comprises a collection of representative use cases for the application of authentication and authorization in constrained environments. These use cases aim at identifying authorization problems that arise during the lifecycle of a constrained device.

We assume that the communication between the devices is based on the Representational State Transfer (REST) architectural style, i.e. a device acts as a server that offers resources such as sensor data and actuators. The resources can be accessed by clients, sometimes without human intervention (M2M). In some situations the communication will happen through intermediaries (e.g. gateways, proxies).

Where specific detail is necessary it is assumed that the devices communicate using CoAP [RFC7252], although most conclusions are generic.

1.1. Terminology

Resource Server (RS): The constrained device which hosts resources the Client wants to access.

Client (C): A device which wants to access a resource on the Resource Server.
This could also be a constrained device.

Resource Owner (RO): The subject who owns the resource and controls its access permissions.

2. Use Cases

This section lists use cases involving constrained devices with certain authorization problems to be solved. Each use case first presents a general description of the application area, then one or more specific use cases, and finally a summary of the authorization-related problems device owners need to be solved.

There are various reasons for assigning a function (client or resource server) to a device, e.g. which device initiates the conversation, how do devices find each other, etc. The definition of the function of a device in a certain use case is not in scope of this document. Readers should be aware that there might be reasons for each setting and that devices might even have different functions at different times.

2.1. Container monitoring

The ability of sensors to communicate environmental data wirelessly opens up new application areas. The use of such sensor systems makes it possible to continuously track and transmit specific characteristics such as temperature, humidity and gas content during the transportation and storage of goods.

The proper handling of the sensors in this scenario is not easy to accomplish. They have to be associated to the appropriate pallet of the respective container. Moreover, the goods and the corresponding sensors belong to specific customers.

During the shipment to their destination the goods often pass stops where they are transloaded to other means of transportation, e.g. from ship transport to road transport.

The transportation and storage of perishable goods is especially challenging since they have to be stored at a constant temperature and with proper ventilation. Additionally, it is very important for the vendors to be informed about irregularities in the temperature and ventilation of fruits to avoid the delivery of decomposed fruits to their customers. The need for a constant monitoring of perishable goods has led to projects such as The Intelligent Container (http://www.intelligentcontainer.com).

2.1.1. Bananas for Munich

A fruit vendor grows bananas in Costa Rica for the German market. It instructs a transport company to deliver the goods via ship to Rotterdam where they are picked up by trucks and transported to a ripening facility. A Munich supermarket chain buys ripened bananas from the fruit vendor and transports them with their own company trucks.

The fruit vendor's quality management wants to assure the quality of their products and thus equips the banana boxes with sensors. The state of the goods is monitored consistently during shipment and ripening and abnormal sensor values are recorded. Additionally, the sensor values are used to control the climate within the cargo containers. Since a wrong sensor value leads to a wrong temperature and thus to spoiled goods, the integrity of the sensor data must be assured.

Due to the high water content of the fruits, the propagation of radio waves is hindered, thus often inhibiting direct communication between nodes [Jedermann14]. Instead, messages are forwarded over multiple hops. Those relaying nodes might belong to different owners. The sensors in the banana boxes cannot always reach the internet during the journey.

The personnel that transloads the goods must be able to locate the goods meant for a specific customer. However the fruit vendor does not want to disclose sensor information pertaining to the condition of the goods to other companies and therefore wants to assure the confidentiality of this data.

When the goods arrive at the supermarket in Munich, the supermarket conducts its own quality check. If no anomalies occurred during the transport, the bananas are admitted for sale.

2.1.2. Authorization Problems Summary

- U1.1 The device owner wants to grant different access rights to a resource to different parties.

- U1.2 The device owner wants to control which devices are allowed to present data to the device.

- U1.3 The device owner wants to grant different access rights for different resources on a device.
- U1.4 The device owner requires the integrity of sensor data.
- U1.5 The device owner requires the confidentiality of sensor data.
- U1.6 The device owner is not always present at the time of access and cannot manually intervene in the authorization process.
- U1.7 The device owner wants to grant temporary access permissions to a party.
- U1.8 Messages between client and resource server might need to be forwarded over multiple hops.
- U1.9 The constrained device might not always be able to reach the internet.

2.2. Home Automation

Automation of the home has the potential to become a big future market for the Internet of Things. A home automation system connects devices in a house to the Internet and thus makes them accessible and manageable remotely. Such devices might control for example heating, ventilation, lighting, home entertainment or home security.

Such a system needs to accommodate a number of regular users (inhabitants, close friends, cleaning personnel) as well as a heterogeneous group of dynamically varying users (visitors, repairmen, delivery men).

As the users are not typically trained in security (or even computer use), the configuration must use secure default settings, and the interface must be well adapted to novice users.

2.2.1. Controlling the Smart Home Infrastructure

Jane and her husband George own a flat which is equipped with home automation devices such as HVAC and shutter control, and they have a motion sensor in the corridor which controls the light bulbs there.

Jane and George can control the shutters and the temperature in each room using either wall-mounted touch panels or their smartphones. Since Jane and George both have a full-time job, they want to be able to change settings remotely, e.g. turn up the heating on a cold day if they will be home earlier than expected.

The couple does not want people in radio range of their devices, e.g. their neighbors, to be able to control them without authorization. Moreover, they don't want burglars to be able to deduce behavioral patterns from eavesdropping on the network.

2.2.2. Seamless Authorization

Jane buys a new light bulb for the corridor and integrates it into the home network (how she does that is not in scope). George is not at home, but Jane wants him to be able to control the new device with his smart phone without the need for additional administration effort.

2.2.3. Remotely letting in a visitor

Jane and George have equipped their home with automated connected door-locks and an alarm system at the door and the windows. The couple can control this system remotely.

Jane and George have invited Jane's parents over for dinner, but are stuck in traffic and can not arrive in time, while Jane's parents who use the subway will arrive punctually. Jane calls her parents and offers to let them in remotely, so they can make themselves comfortable while waiting.

Jane's parents download an application that lets them communicate with Jane's door-lock and alarm system. Then Jane sets temporary permissions that allow them to open the door, and shut down the alarm when they arrive.

The security system controlling the door-locks and alarm system needs to be at least as secure as for a comparable unautomated home.

2.2.4. Authorization Problems Summary

- o U2.1 A home owner wants to spontaneously provision authorization means to visitors.

- o U2.2 A home owner wants to spontaneously change the home's access control policies.

- o U2.3 A home owner wants to apply different access rights for different users.

- o U2.4 A home owner wants to apply context-based conditions (presence, time) to authorizations, and the devices need to be able to verify these conditions.

- U2.5 The smart home devices need to be able to communicate with different control devices (e.g. wall-mounted touch panels, smartphones, electronic key fobs).

- U2.6 The access control configuration of the automated home needs to be secure by default.

- U2.7 The access control policies need to be easy to edit, even remotely and it needs to be easy to get access with correct authorization.

- U2.8 The owners of the automated home wants to prevent eavesdroppers form being able to deduce behavioral profiles from the home network.

- U2.9 Usability is particularly important in this scenario since administrative tasks such as installation, configuration and decommissioning of devices likely need to be performed by the home owners who in most cases have little knowledge of security.

- U2.10 Home Owners want their devices to seamlessly (and in some cases even unnoticeably) fulfill their purpose. The administration effort needs to be kept at a minimum.

2.3. Personal Health Monitoring

The use of wearable health monitoring technology is expected to grow strongly, as a multitude of novel devices are developed and marketed. The need for open industry standards to ensure interoperability between products has lead to initiatives such as Continua Alliance (continuaalliance.org) and Personal Connected Health Alliance (pchalliance.org). Personal health devices are typically battery driven, and located physically on the user. They monitor some bodily function, such as e.g. temperature, blood pressure, or pulse. They are connected to the Internet through an intermediary base-station, using wireless technologies. Through this connection they report the monitored data to some entity, which may either be the user herself, or some medical personnel in charge of the user.

Medical data has always been considered as very sensitive, and therefore requires good protection against unauthorized disclosure. A frequent, conflicting requirement is the capability for medical personnel to gain emergency access, even if no specific access rights exist. As a result, the importance of secure audit logs increases in such scenarios.

Since the users are not typically trained in security (or even computer use), the configuration must use secure default settings, and the interface must be well adapted to novice users. Parts of the system must operate with minimal maintenance. Especially frequent changes of battery are unacceptable.

2.3.1. John and the heart rate monitor

John has a heart condition, that can result in sudden cardiac arrests. He therefore uses a device called HeartGuard that monitors his heart rate and his position. In case of a cardiac arrest it automatically sends an alarm to an emergency service, transmitting John's current location. The HeartGuard also broadcasts emergency information in the neighborhood to notify doctors or people with certain skills who have been enrolled in an emergency program, e.g. people who got training in heart and lung rescue. For doctors, medical information or diagnosis can be provided with the notification to improve immediate treatment.

The device includes some smart logic, with which it identifies its owner John and allows him to configure the device's settings, including access control.
This prevents situation where someone else wearing that device can act as the owner and mess up the access control and security settings.

John can configure additional persons that get notified in an emergency, for example his daughter Jill. Furthermore the device stores data on John's heart rate, which can later be accessed by a physician to assess the condition of John's heart.

However John is a rather private person, and is worried that Jill might use HeartGuard to monitor his location while there is no emergency. Furthermore he doesn't want his health insurance to get access to the HeartGuard data, or even to the fact that he is wearing a HeartGuard, since they might refuse to renew his insurance if they decided he was too big a risk for them.

NOTE: Monitoring of some state parameter (e.g. an alarm button) and the position of a person also fits well into an elderly care service. This is particularly useful for people suffering from dementia, where the relatives or caregivers need to be notified of the whereabouts of the person under certain conditions. In this case it is not the patient that decides about access.

2.3.2. Authorization Problems Summary

- U3.1 A device owner wants to pre-configure access rights to specific data for persons or groups, in the context of an emergency.

- U3.2 A device owner wants to selectively allow different persons or groups to access medical data.

- U3.3 A device owner wants to block access to specific persons in an otherwise allowed group (e.g. doctors in an emergency), if he mistrusts them.

- U3.4 The security measures could affect battery lifetime of the devices and should changes of battery are highly inconvenient.

- U3.5 Devices are often used with default access control settings.

- U3.6 Device users are often not trained in computer use and especially computer security.

- U3.7 Security mechanisms themselves could provide opportunities for denial of service attacks on the device.

- U3.8 The device provides a service that can be fatal for the device owner if it fails. Accordingly, the device owner wants a security mechanism to provide a high level of security.

2.4. Building Automation

Buildings for commercial use such as shopping malls or office buildings nowadays are equipped increasingly with semi-automatic components to enhance the overall living quality and to save energy where possible. This includes for example heating, ventilation and air condition (HVAC) as well as illumination and security systems such as fire alarms.

Different areas of these buildings are often exclusively leased to different companies. However they also share some of the common areas of the building.
Accordingly, a company must be able to control the light and HVAC system of its own part of the building and must not have access to control rooms that belong to other companies.

Some parts of the building automation system such as entrance illumination and fire alarm systems are controlled either by all parties together or by a service company.

2.4.1. Device Lifecycle

2.4.1.1. Installation and Commissioning

A building is hired out to different companies for office space. This building features various automated systems, such as a fire alarm system, which is triggered by several smoke detectors which are spread out across the building. It also has automated HVAC, lighting and physical access control systems.

A vacant area of the building has been recently leased to company A. Before moving into its new office, Company A wishes to replace the lighting with a more energy efficient and a better light quality luminaries. They hire an installation and commissioning company C to redo the illumination. Company C is instructed to integrate the new lighting devices, which may be from multiple manufacturers, into the existing lighting infrastructure of the building which includes presence sensors, switches, controllers etc.

Company C gets the necessary authorization from the service company to interact with the existing Building and Lighting Management System (BLMS). To prevent disturbance to other occupants of the building, Company C is provided authorization to perform the commissioning only during non-office hours and only to modify configuration on devices belonging to the domain of Company A's space. After installation (wiring) of the new lighting devices, the commissioner adds the devices into the company A's lighting domain.

Once the devices are in the correct domain, the commissioner authorizes the interaction rules between the new lighting devices and existing devices like presence sensors. For this, the commissioner creates the authorization rules on the BLMS which define which lights form a group and which sensors /switches/controllers are allowed to control which groups. These authorization rules may be context based like time of the day (office or non-office hours) or location of the handheld lighting controller etc.

2.4.1.2. Operational

Company A's staff move into the newly furnished office space. Most lighting is controlled by presence sensors which control the lighting of specific group of lights based on the authorization rules in the BLMS. Additionally employees are allowed to manually override the lighting brightness and color in their office by using the switches or handheld controllers. Such changes are allowed only if the authorization rules exist in the BLMS. For example lighting in the corridors may not be manually adjustable.

At the end of the day, lighting is dimmed down or switched off if no occupancy is detected even if manually overridden during the day.

On a later date company B also moves into the same building, and shares some of the common spaces with company A. On a really hot day James who works for company A turns on the air condition in his office. Lucy who works for company B wants to make tea using an electric kettle. After she turned it on she goes outside to talk to a colleague until the water is boiling. Unfortunately, her kettle has a malfunction which causes overheating and results in a smoldering fire of the kettle's plastic case.

Due to the smoke coming from the kettle the fire alarm is triggered. Alarm sirens throughout the building are switched on simultaneously (using a broadcastor multicast) to alert the staff of both companies. Additionally, the ventilation system of the whole building is closed off to prevent the smoke from spreading and to withdraw oxygen from the fire. The smoke cannot get into James' office although he turned on his air condition because the fire alarm overrides the manual setting by sending commands (broadcast or multicast) to switch off all the air conditioning.

The fire department is notified of the fire automatically and arrives within a short time. After inspecting the damage and extinguishing the smoldering fire a fire fighter resets the fire alarm because only the fire department is authorized to do that.

2.4.1.3. Maintenance

Company A's staff are annoyed that the lights switch off too often in their rooms if they work silently in front of their computer. Company A notifies the commissioning Company C about the issue and asks them to increase the delay before lights switch off.

Company C again gets the necessary authorization from the service company to interact with the BLMS. The commissioner's tool gets the necessary authorization from BMLS to send a configuration change to all lighting devices in Company A's offices to increase their delay before they switch off.

2.4.1.4. Decommissioning

Company A has noticed that the handheld controllers are often misplaced and hard to find when needed. So most of the time staff use the existing wall switches for manual control. Company A decides it would be better to completely remove handheld controllers and asks Company C to decommission them from the lighting system.

Company C again gets the necessary authorization from the service company to interact with the BLMS. The commissioner now deletes any rules that allowed handheld controllers authorization to control the lighting. Additionally the commissioner instructs the BLMS to push these new rules to prevent cached rules at the end devices from being used.

2.4.2. Authorization Problems Summary

- U4.1 Device owners want to be able to add a new device to their administrative domain (commissioning).

- U4.2 Device owners want to be able to integrate a device that formerly belonged to a different administrative domain to their own administrative domain (handover).

- U4.3 Device owner want to be able to remove a device from their administrative domain (decomissioning).

- U4.4 Device owners want to be able to delegate selected administration tasks for their devices to others.

- U4.5 The device owner wants to be able to define context-based Authorization rules.

- U4.6 The device owner wants to be able to revoke granted permissions and delegations.

- U4.7 The device owner wants to allow only authorized access to device resources (default deny).

- U4.8 The device owner wants to be able to authorize a device to control several devices at the same time using a multicast protocol.

- U4.9 Device owners want to be able to interconnect their own subsystems with those from a different operational domain while keeping the control over the authorizations (e.g. granting and revoking permissions) for their devices.

2.5. Smart Metering

Automated measuring of customer consumption is an established technology for electricity, water, and gas providers. Increasingly these systems also feature networking capability to allow for remote management. Such systems are in use for commercial, industrial and residential customers and require a certain level of security, in order to avoid economic loss to the providers, vulnerability of the distribution system, as well as disruption of services for the customers.

The smart metering equipment for gas and water solutions is battery driven and communication should be used sparingly due to battery consumption. Therefore the types of meters sleep most of the time, and only wake up every minute/hour to check for incoming instructions. Furthermore they wake up a few times a day (based on their configuration) to upload their measured metering data.

Different networking topologies exist for smart metering solutions. Based on environment, regulatory rules and expected cost, one or a mixture of these topologies may be deployed to collect the metering information. Drive-By metering is one of the most current solutions deployed for collection of gas and water meters.

2.5.1. Drive-by metering

A service operator offers smart metering infrastructures and related services to various utility companies. Among these is a water provider, who in turn supplies several residential complexes in a city. The smart meters are installed in the end customer’s homes to measure water consumption and thus generate billing data for the utility company. The meters do so by sending data to a base station. Several base stations are installed around the city to collect the metering data. However in the denser urban areas, the base stations would have to be installed very close to the meters. This would require a high number of base stations and expose this more expensive equipment to manipulation or sabotage. The service operator has therefore chosen another approach, which is to drive around with a mobile base-station and let the meters connect to that in regular intervals in order to gather metering data.

2.5.2. Meshed Topology

In another deployment, the water meters are installed in a building that already has power meters installed, the latter are mains powered, and are therefore not subject to the same power saving restrictions. The water meters can therefore use the power meters as proxies, in order to achieve better connectivity. This requires the security measures on the water meters to work through intermediaries.

2.5.3. Advanced Metering Infrastructure

A utility company is updating its old utility distribution network with advanced meters and new communication systems, known as an Advanced Metering Infrastructure (AMI). AMI refers to a system that measures, collects and analyzes usage, and interacts with metering devices such as electricity meters, gas meters, heat meters, and water meters, through various communication media either on request (on-demand) or on pre-defined schedules. Based on this technology, new services make it possible for consumers to control their utility consumption and reduce costs by supporting new tariff models from utility companies, and more accurate and timely billing.

The technical solution is based on levels of data aggregation between smart meters located at the consumer premises and the Meter Data Management (MDM) system located at the utility company. Two possible intermediate levels are:

o Head-End System (HES) which is hardware and software that receives the stream of meter data and exposes an interface to the MDM.

o Data Collection (DC) units located in a local network communicating with a number of smart meters and with a backhaul interface communicating with the HES, e.g. using cellular communication.

For reasons of efficiency and cost end-to-end connectivity is not always feasible, so metering data is stored in batches in DC for some time before being forwarded to the HES, and in turn accessed by the MDM. The HES and the DC units may be operated by a third party service operator on behalf of the utility company. One responsibility of the service operator is to make sure that meter readings are performed and delivered to the HES. An example of a Service Level Agreement between the service operator and the utility company is e.g. "at least 95 % of the meters have readings recorded during the last 72 hours".

2.5.4. Authorization Problems Summary

o U5.1 Devices are installed in hostile environments where they are physically accessible by attackers. Device owners want to make sure that an attacker cannot use a captured device to attack other parts of their infrastructure.

o U5.2 Device owners want to restrict which entities are allowed to write data to the devices and thus ensure the integrity of the data on their devices.

o U5.3 The device owner wants to control which entities are allowed to read data on the devices and protect such data in transfer.

o U5.4 The devices may have intermittent Internet connectivity.

- U5.5 The device owner is not always present at the time of access and cannot manually intervene in the authorization process.
- U5.6 When authorization policies are updated it is impossible, or at least very inefficient to contact all affected devices directly.
- U5.7 Messages between a client and the device may need to be stored and forwarded over multiple nodes.

2.6. Sports and Entertainment

In the area of leisure time activities, applications can benefit from the small size and weight of constrained devices. Sensors and actuators with various functionalities can be integrated into fitness equipment, games and even clothes. Owners can carry their devices around with them at all times.

Usability is especially important in this area since owners will often want to spontaneously interconnect their devices with others. Therefore the configuration of access permissions must be simple and fast and not require much effort at the time of access (preferably none at all).

The required level of security will in most cases be low since security breaches will likely have less severe consequences. The continuous monitoring of data might however enable an attacker to create behavioral or movement profiles. Moreover, the aggregation of data can seriously increase the impact on the privacy of device owners.

2.6.1. Dynamically Connecting Smart Sports Equipment

Jody is a an enthusiastic runner. To keep track of her training progress, she has smart running shoes that measure the pressure at various points beneath her feet to count her steps, detect irregularities in her stride and help her to improve her posture and running style. On a sunny afternoon, she goes to the Finnbahn track near her home to work out. She meets her friend Lynn who shows her the smart fitness watch she bought a few days ago. The watch can measure the wearer’s pulse, show speed and distance, and keep track of the configured training program. The girls detect that the watch can be connected with Jody’s shoes and then can additionally display the information the shoes provide.

Jody asks Lynn to let her try the watch and lend it to her for the afternoon. Lynn agrees but doesn’t want Jody to access her training plan. She configures the access policies for the watch so that

Jody’s shoes are allowed to access the display and measuring features but cannot read or add training data. Jody’s shoes connect to Lynn’s watch after only a press of a button because Jody already configured access rights for devices that belong to Lynn a while ago.

After an hour, Jody gives the watch back and both girls terminate the connection between their devices.

2.6.2. Authorization Problems Summary

- U6.1 The owner of a device wants to be able to grant access rights dynamically when needed.

- U6.2 The owner wants the configuration of access rights to work with very little effort.

- U6.3 The device owner wants to be able to preconfigure access policies that grant certain access permissions to devices with certain attributes (e.g. devices of a certain user) without additional configuration effort at the time of access.

- U6.4 Device owners wants to protect the confidentiality of their data for privacy reasons.

- U6.5 Devices might not have an Internet connection at the time of access.

2.7. Industrial Control Systems

Industrial control systems (ICS) and especially supervisory control and data acquisition systems (SCADA) use a multitude of sensors and actuators in order to monitor and control industrial processes in the physical world. Example processes include manufacturing, power generation, and refining of raw materials.

Since the advent of the Stuxnet worm it has become obvious to the general public how vulnerable this kind of systems are, especially when connected to the Internet. The severity of these vulnerabilities are exacerbated by the fact that many ICS are used to control critical public infrastructure, such as power, water treatment of traffic control. Nevertheless the economical advantages of connecting such systems to the Internet can be significant if appropriate security measures are put in place.

2.7.1. Oil Platform Control

An oil platform uses an industrial control system to monitor data and control equipment. The purpose of this system is to gather and process data from a large number of sensors, and control actuators such as valves and switches to steer the oil extraction process on the platform. Raw data, alarms, reports and other information are also available to the operators, who can intervene with manual commands. Many of the sensors are connected to the controlling units by direct wire, but the operator is slowly replacing these units by wireless ones, since this makes maintenance easier.

The controlling units are connected to the Internet, to allow for remote administration, since it is expensive and inconvenient to fly in a technician to the platform.

The main interest of the operator is to ensure the integrity of control messages and sensor readings. The access to some resources needs to be restricted to certain clients, e.g. the operator wants wireless actuators only to accept commands by authorized control units.

The owner of the platform also wants to collect auditing information for liability reasons.

2.7.2. Authorization Problems Summary

o U7.1 The device owner wants to ensure that only authorized clients can read data from sensors and sent commands to actuators.

o U7.2 The device owner wants to ensure that data coming from sensors and commands sent to actuators are authentic.

o U7.3 Some devices do not have direct Internet connection.

o U7.4 Some devices have wired connection while other use wireless.

o U7.5 The execution of unauthorized commands in an ICS can lead to significant financial damage, and threaten the availability of critical infrastructure services. Accordingly, the device owner wants a security solution that provides a very high level of security.

3. Security Considerations

As the use cases listed in this document demonstrate, constrained devices are used in various application areas. The appeal of these devices is that they are small and inexpensive. That makes it easy to integrate them into many aspects of everyday life. Therefore, the devices will be entrusted with vast amounts of valuable data or even control functions, that need to be protected from unauthorized access.

Moreover, the aggregation of data must be considered: attackers might not only collect data from a single device but from many devices, thus increasing the potential damage.

Not only the data on the constrained devices themselves is threatened, the devices might also be abused as an intrusion point to infiltrate a network. Once an attacker gained control over the device, it can be used to attack other devices as well. Due to their limited capabilities, constrained devices appear as the weakest link in the network and hence pose an attractive target for attackers.

This section summarizes the security problems highlighted by the use cases above and provides guidelines for the design of protocols for authentication and authorization in constrained RESTful environments.

3.1. Attacks

This document lists security problems that owners of constrained devices want to solve. Further analysis of attack scenarios is not in scope of the document. However, there are attacks that must be considered by solution developers.

Because of the expected large number of devices and their ubiquity, constrained devices increase the danger from Pervasive Monitoring [RFC7258] attacks.

As some of the use cases indicate, constrained devices may be installed in hostile environments where they are physically accessible (see Section 2.5). Protection from physical attacks is not in the scope of ACE, but should be kept in mind by developers of authorization solutions.

Denial of service (DoS) attacks threaten the availability of services a device provides. E.g., an attacker can induce a device to perform steps of a heavy weight security protocol (e.g. Datagram Transport Layer Security (DTLS) [RFC6347]) before authentication and authorization can be verified, thus exhausting the device's system resources. This leads to a temporary or - e.g. if the batteries are drained - permanent failure of the service. For some services of constrained devices, availability is especially important (see Section 2.3). Because of their limitations, constrained devices are especially vulnerable to denial of service attacks. Solution designers must be particularly careful to consider these limitations in every part of the protocol. This includes:

- o Battery usage

- o Number of message exchanges required by security measures

- Size of data that is transmitted (e.g. authentication and access control data)

- Size of code required to run the protocol

- Size of RAM memory and stack required to run the protocol

Another category of attacks that needs to be considered by solution developers is session interception and hijacking.

3.2. Configuration of Access Permissions

- The access control policies of the Resource Owner need to be enforced (all use cases): The access control policies set by the Resource Owner need to be provisioned to the device that enforces the authorization and applied to every incoming request.

- A single resource might have different access rights for different requesting entities (all use cases).

  Rationale: In some cases different types of users need different access rights, as opposed to a binary approach where the same access permissions are granted to all authenticated users.

- A device might host several resources where each resource has its own access control policy (all use cases).

- The device that makes the policy decisions should be able to evaluate context-based permissions such as location or time of access (see e.g. Section 2.2, Section 2.3, Section 2.4). Access may depend on local conditions, e.g. access to health data in an emergency. The device that makes the policy decisions should be able to take such conditions into account.

3.3. Design Considerations for Authorization Solutions

- Devices need to be enabled to enforce the owner’s authorization policies without the owner’s intervention at the time of the access request (see e.g. Section 2.1, Section 2.2, Section 2.4, Section 2.5).

- Authorization solutions need to consider that constrained devices might not have internet access at the time of the access request (see e.g. Section 2.1, Section 2.3, Section 2.5, Section 2.6).

- It should be possible to update access control policies without manually re-provisioning individual devices (see e.g. Section 2.2, Section 2.3, Section 2.5, Section 2.6).

Rationale: Peers can change rapidly which makes manual re-provisioning unreasonably expensive.

- o Owners might define authorization policies for a large number of devices that might only have intermittent connectivity. Distributing policy updates to every device for every update might not be a feasible solution.

- o It must be possible to dynamically revoke authorizations (see e.g. Section 2.4).

- o The authentication and access control protocol can put undue burden on the constrained resources of a device participating in the protocol. An authorization solutions must take the limitations of the constrained devices into account (see also Section 3.1).

- o Secure default settings are needed for the initial state of the authentication and authorization protocols (all use cases).

  Rationale: Many attacks exploit insecure default settings, and experience shows that default settings are frequently left unchanged by the end users.

- o Access to resources on other devices should only be permitted if a rule exists that explicitly allows this access (default deny).

- o Usability is important for all use cases. The configuration of authorization policies as well as the gaining access to devices must be simple for the users of the devices. Special care needs to be taken for home scenarios where access control policies have to be configured by users that are typically not trained in security (see Section 2.2, Section 2.6).

3.4. Proxies

In some cases, the traffic between Client and Resource Server might go through intermediary nodes (e.g. proxies, gateways). This might affect the function or the security model of authentication and access control protocols e.g. end-to-end security between Client and Resource Server with DTLS might not be possible (see Section 2.5).

4. Privacy Considerations

Many of the devices that are in focus of this document register data from the physical world (sensors) or affect processes in the physical world (actuators), which may involve data or processes belonging to individuals. To make matters worse the sensor data may be recorded continuously thus allowing to gather significant information about an individual subject to the sensor readings. Therefore privacy protection is especially important, and Authentication and Access control are important tools for this, since they make it possible to control who gets access to private data.

Privacy protection can also be weighted in when evaluating the need for end-to-end confidentiality, since otherwise intermediary nodes will learn the content of potentially sensitive messages sent between a client and a resource server and thereby endanger the privacy of the individual that may be subject of this data.

In some cases, even the possession of a certain type of device can be confidential, e.g. owners might not want to others to know that they are wearing a certain medical device (see Section 2.3).

The personal health monitoring use case (see Section 2.3) indicates the need for secure audit logs which impose specific requirements on a solution. Auditing is not in the scope of ACE. However, if an authorization solution provides means for audit logs, it must consider the impact of logged data for the privacy of the owner and other parties involved.
Suitable measures for protecting and purging the logs must be taken during operation, maintenance and decommissioning of the device.

5. Acknowledgments

The authors would like to thank Olaf Bergmann, Sumit Singhal, John Mattson, Mohit Sethi, Carsten Bormann, Martin Murillo, Corinna Schmitt, Hannes Tschofenig, Erik Wahlstroem, and Andreas Backman for reviewing and/or contributing to the document. Also, thanks to Markus Becker, Thomas Poetsch and Koojana Kuladinithi for their input on the container monitoring use case.

6. IANA Considerations

This document has no IANA actions.

7. Informative References

[Jedermann14]
Jedermann, R., Poetsch, T., and C. LLoyd, "Communication techniques and challenges for wireless food quality monitoring", Philosophical Transactions of the Royal Society A Mathematical, Physical and Engineering Sciences, May 2014.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, May 2014.

Authors' Addresses

Ludwig Seitz (editor)
SICS Swedish ICT AB
Scheelevaegen 17
Lund 223 70
Sweden

Email: ludwig@sics.se

Stefanie Gerdes (editor)
Universitaet Bremen TZI
Postfach 330440
Bremen 28359
Germany

Phone: +49-421-218-63906
Email: gerdes@tzi.org

Goeran Selander
Ericsson
Faroegatan 6
Kista 164 80
Sweden

Email: goran.selander@ericsson.com

Mehdi Mani
Itron
52, rue Camille Desmoulins
Issy-les-Moulineaux 92130
France

Email: Mehdi.Mani@itron.com

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

Email: sandeep.kumar@philips.com

```
Network Working Group                                      H. Tschofenig
Internet-Draft                                                  ARM Ltd.
Intended status: Best Current Practice                 February 14, 2014
Expires: August 18, 2014
```

Authentication and Authorization for Constrained Environments (ACE): Overview of Existing Security Protocols
draft-tschofenig-ace-overview-00.txt

Abstract

This document surveys existing three party authentication and authorization protocols for use with Internet of Things use cases. The discussed protocol frameworks are Kerberos, OAuth, ABFAB, and the certificate model. The aim is to understand whether any of the available standardized security protocols are re-usable for constrained environments. A future version of this document will provide a more detailed analysis against the requirements.

Status of This Memo

Copyright Notice

1. Introduction

[I-D.seitz-ace-usecases] introduces a number of use cases that require device-to-device authentication whereby both devices may be constrained. [I-D.ietf-lwig-terminology] discusses the different types of constraints of these devices.

This document aims to raise the high-level question about the possible re-use of existing three party authentication and key exchange protocols for use in IoT environments. This version of the document does not aim to map requirements derived from the use cases against these protocols. Such a detailed analysis is premature at this point when use case descriptions are still in flux.

The starting assumption for the architectures in this document is that a device (a client) wants to access some resource (referred as service in this document). It unfortunately does not have any relationship with the server offering that service. Figure 1 shows the scenario graphically.

```
        +------------+    no prior        +------------+
        |   Client   |    relationship    |  Service   |
        |            |                    |            |
        +------------+                    +------------+
```

Figure 1: Two Party Scenario.

Imagine that the client is a light-switch and the service is a light-bulb.

Today, companies solve this case by using a pairing protocol (at the link layer typically) where the two devices execute a special imprinting/pairing protocol to establish an initial key by using out-of-band (OOB) channel. This OOB channel can come in many forms:

- o Using an alternative communication channel, such as a USB stick, Ethernet cable

- o Human involvement by comparing hashed keys, entering passkeys, scanning QR codes

- o Second wireless connectivity (e.g., infrared) o Proximity-based information

The pairing is a suitable approach where wireless communication replaces a wired communication technology previously used. For example, a headset connected to a music player using a wired connection is replaced with the wireless version. Not all use cases do, however, allow users to pair their device with other devices upfront. Consider an enterprise with electronic door locks. It is hard to imagine an employee who has to pair their digital key with every door in the building first before they use the system.

Requiring every device to pair with every other device upfront is often inconvenient or not feasible. Hence, this document does not explore pairing solutions further. To offer an improved user experience with better scalability properties a device might either share credentials with some trusted third party. There are various ways how credentials can be shared with these trusted third parties. For example, credentials may be provisioned during the manufacturing process or devices may have been paired with the trusted third party (in case the trusted third party is local to the user). In fact, today is it very common for IoT devices to have at least credentials pre-provisioned for use with the vendor / manufacturer of the device to allow software updates to be provided securely.

Thus, we move to a model where the device (client) shares some credentials with a trusted third party. This trusted third party does not need to be a server on the Internet; ideally it could also be operated locally within someones' home, within an enterprise, or within a factory.

This three party architecture is shown in Figure 2.

```
                                          +-----------+
                                          |  Trusted  |
             +---------------------------->|Third Party|
             |        Key                 +-----------+
             |                                  ^
             |                                  |
             |                                  |
             |                                  | Key
             |                                  |
             |                                  |
             |                                  |
             v                                  v
        +-----------+  no prior           +-----+-----+
        |           |  relationship       |           |
        |   Client  | <.................>| Service   |
        +-----------+                     +-----------+
```

Figure 2: Three Party Scenario.

This three party architecture and messaging pattern has been explored with prior IETF work and this document lists the most relevant efforts (on a high level).

The goal of the communication exchange is that the client has been authorized to access the service, and is able to secure the exchange of information. The client and the service may, optionally, possess keying material for future use of the service with the benefit of better performance for future interactions.

Note: This document does not aim to cover the use cases in their entirety. First, we assume that the security protocol interaction for link layer authentication is outside the scope. The focus of this document is on the application layer interactions when accessing services. Second, this document does not survey access control policy languages and mechanisms for managing these access control policies. These policies are important since many of the systems described below only provide an answer to the question 'Who is the holder of this key?' and standards for answering the question 'Can this key be used for this purpose?' (authorization) are often realized in a proprietary way.

While Figure 1 shows three parties the protocols described in Section 2 have been generalized to four or even multi-party scenario. The result is shown in Figure 2.

```
   +-----------+                         +-----------+
   |  Trusted  |   Agreement w/key       | Trusted   |
   |  Third    |<----------------------->| Third     |
   |  Party A  |                         | Party B   |
   +-----------+                         +-----------+
         ^                                     ^
         |                                     |
         |  Key                                |  Key
         |                                     |
         |                                     |
         |                                     |
         |                                     |
         v                                     v
    +-----------+  no prior              +-----+-----+
    |           |  relationship          |           |
    |   Client  | <....................>| Service   |
    +-----------+                        +-----------+
```

Figure 3: Generalization of Three Party Scenario.

2. Three Party Security Frameworks

This section introduces four authentication and authorization frameworks standardized in the IETF. The description is intentionally kept at a high level and a reader is encouraged to consult the referenced documents for details and various options these protocols offer. The terminology with each of these protocols is lightly different and appropriate mappings have been applied.

To demonstrate the level of maturity of these frameworks availability of products, source code, and deployment experience is mentioned for each of these frameworks. Note, however, that this experience does not imply suitability for use with the IoT environment.

2.1. Application Bridging for Federated Access Beyond Web Architecture

This section describes the Application Bridging for Federated Access Beyond Web architecture [I-D.ietf-abfab-arch], which builds on the Authentication, Authorization and Accounting (AAA) framework. The AAA framework re-uses the Extensible Authentication Protocol (EAP) [RFC5247] and EAP methods for the authentication protocol capabilities. A detailed description of the AAA keying framework can be found in [RFC5247].

```
                                       +---------------+
                                       |   Identity    |
                                       |   Provider    |
                                       |    (IdP)      |
                                       +-^-----------^-+
                                         * EAP       o RADIUS
                                         *           o
                                         *           o
      +-------------+                  +-v-----------v--+
      |             |                  |                |
      | Client      |  EAP/EAP Method  | Relying        |
      |             |<****************>| Party          |
      |             |  GSS-API         |                |
      |             |<---------------->|                |
      |             |  Application     |                |
      |             |  Data            |                |
      |             |<================>|                |
      +-------------+                  +----------------+

   Terminology Mapping:
    - The term 'Relying Party' corresponds to the 'service'.
    - The term 'Identity Provider' corresponds to the
      'trusted third party'.
```

Figure 4: ABFAB Architecture.

With the message exchange shown in Figure 4 the client wants to obtain access to a service and starts interacting with that service. Since no prior relationship between the client and the service is assumed the EAP message exchanges is relayed by the service and the EAP server component of the IdP. Between the client and the service these EAP payloads are encapsulated within the GSS-API. After a successful authentication and authorization session keys are delivered from the IdP to the service and can then be used to secure the application layer data exchange between the client and the service.

While the use of EAP and the AAA architecture has mostly found use for network access authentication the work on ABFAB applies this architecture to application layer services.

Pros:

o Re-uses existing protocols: RADIUS, GSS-API, EAP, EAP methods.

- Security properties of the AAA / EAP framework well studied and large deployments of the AAA framework exist.

- Products and open source code exists for EAP, EAP methods, RADIUS, and the GSS-API. The extensions needed for ABFAB also have been implemented but they are less mature compared to the EAP/AAA deployment.

- Large range of EAP methods available offering all possible authentication and key exchange protocols for authentication between the client and the AAA server. These mechanisms have been deployed and are in widespread use today. While many EAP methods have been standardized only a few are in widespread use in non-IoT environments. However, there are many (open source) implementations available such that further experience concerning IoT suitability can be gathered.

- IoT devices might use the AAA/EAP architecture for network access authentication (e.g., WLAN-based, IEEE 802.15.4-based ZigBee-IP deployments).

- The AAA framework also supports authentication in a federated environment.

- Authorization information is conveyed within RADIUS (and potentially in SAML assertions, as envisioned by ABFAB).

Cons:

- The initial authentication and authorization exchange requires real-time interaction between the AAA server and the service.

- Deployments have so far used this architecture mainly for network access and for specific applications (VoIP) only. Experience with other applications, as ABFAB envisions, is rather limited.

- ABFAB architecture uses layering of EAP within the GSS-API, which adds additional overhead. A binding for the transport of EAP payloads in CoAP, for example, does not exist.

- No unified authorization policy language has been defined for the AAA/EAP architecture. Instead, RADIUS attributes carry information about access control decisions.

2.2. Kerberos

Kerberos [RFC4120] is authentication system for distributed environments that has enjoyed deployment for more than three decades. The security properties have been extensively studied and various implementations exist.

```
                  +----------------+
                  | Authentication |
                  | Server (AS)    |
                  +----------------+
                  ^        /
       Request   /        /
       Ticket   /        /
               /        /Ticket
              /        /{SK}C-KDC
             /        /
            /        /
           /        /
          /        v
        +-----------+                             +-----------+
        |           |   Ticket + Authenticator    |           |
        |  Client   |---------------------------->|  Server   |
        |           |<===========================>|           |
        +-----------+    Application Data         +-----------+

   Terminology Mapping:
     - The term AS corresponds to the 'trusted third party.'
     - The term Server corresponds to the 'service'.
```

Figure 5: Kerberos.

The Kerberos exchange shown in Figure 5 illustrates a client who wants to get access to a server. It first has to interact with the Authentication Server (AS) to request a ticket. In response, the AS provides a ticket, which is a data structure encrypted with a key known only between the server and the AS. This ticket includes information about the client, a session key (SK) for later use, and various other security relevant information elements. The client also obtains the session key encrypted with a key it shares with the AS.

When a service access is required then the client interacts with the server and presents the ticket along with an Authenticator. The Authenticator demonstrates that the client was able to decrypt the session key with the key it shares with the AS and that it was able to apply this key to compute a keyed message digest over several fields, including a time-stamp, when accessing the service. The time-stamp avoids replay attacks.

Pros:

- Re-uses existing protocol: Kerberos

- Security properties well studied and large deployments exist.

- Products and open source service exist.

- Most parts of Kerberos, particularly the ticket concept, are designed with symmetric key cryptography, which improves performance. The Kerberos ticket is consequently fairly small and uses a binary encoding.

- Kerberos also supports cross-realm authentication for scalable deployments.

- Kerberos also specifies a UDP-based transport.

- The message exchanges between the client and the service can be tailored to the need of the application.

Cons:

- Each ticket is only usable for a single service (intentionally). As such, new tickets have to be requested whenever the client wants to access a new service or when the ticket expired.

- For the authentication between the client and the KDC a limited number of authentication protocols have been specified.

- Kerberos uses ASN.1 for encoding of the ticket and various messages. This may increase implementation complexity but the binary encoding is more efficient than other encodings, like XML or JSON.

- No standardized access control policy has been standardized for inclusion inside a ticket. Proprietary policies are, however, used in real-world deployments.

- A CoAP binding for the KRB_PRIV and the KRB_SAFE message exchanges used to secure application data between the client and the service have not been defined.

2.3. OAuth

The OAuth protocol is a recent development for the Web, which re-uses the Kerberos interaction pattern with influences from the Web / mobile app space. It initially aimed to solve the problem of delegated access to protected resources where websites asked users to share their long-term password. Over time OAuth has been used in other use cases that require delegated access.

```
                            +-------------+
                            |Authorization|
                            |Server (AS)  |
                            +-------------+
                            ^        /
                 Request   /        /
                 Access   /        /
                 Token   /        /Access Token
                        /        /
                       /        /
                      /        /
                     /        /
      O             /        v
     /|\          +-----------+                  +-----------+
      |    -----> |           |   Access Token   | Resource  |
     / \   <----- |  Client   |----------------->|  Server   |
   Resource       |           |<================>|   (RS)    |
    Owner         +-----------+ Application Data +-----------+

   Terminology Mapping:
     - The term AS corresponds to the 'trusted third party'.
     - The term RS corresponds to the 'service'.
```

Figure 6: Simplified OAuth Architecture.

Figure 6 shows the high-level OAuth message exchange. The canonical OAuth example allows a web user (resource owner) to grant a printing service (client) access to her private photos stored at a photo sharing service (resource server), without sharing her username and password. Instead, she authenticates directly with the authorization server which issues the printing service delegation-specific credentials.

Pros:

- o Re-uses existing protocols: OAuth Core [RFC6749], OAuth Bearer Token [RFC6750] OAuth MAC Token [I-D.ietf-oauth-v2-http-mac]/ HOTK [I-D.tschofenig-oauth-hotk], JWT [I-D.ietf-oauth-json-web-token].

- o Large deployments in the Web environment exist, which use the OAuth Bearer Token.

- o Products and open source service exist.

- o OAuth is flexible with regard to the used cryptography. A standardized format for the access token has been described with the JSON Web Token (JWT). For security protection of the JWT various specifications from the JOSE working group are available.

- o The message exchanges between the client and the service can be tailored to the need of the application. Bindings are available for HTTPS and SASL [I-D.ietf-kitten-sasl-oauth].

- o With regard to the offered security mechanism the interaction between the client and the resource server gives several choices: The OAuth Bearer Token requires a TLS exchange between the client and the resource server. The MAC Token specification is conceptually similar to Kerberos; a version based on asymmetric cryptography exists as well (see HOTK).

Cons:

- o For an environment with more than one authorization server or where the authorization server is located in a different domain than the resource server the standardization work is still in progress. Efforts have mostly be done in Kantara with the User-Managed-Access (UMA) working group.

- o A binding for CoAP does not exist for the client to authorization server nor for the client to resource server.

- o The OAuth architecture does not standardize the authentication procedure of the resource owner to the authorization server itself. This is a common approach for the Web environment where a number of different authentication protocols are in use in the browser. As such, the protocol works with any authentication mechanism.

2.4. Certificate Model

Prior work on the Public Key Infrastructure, certificate formats, certificate extensions, and various certificate management protocols can be re-used in the IoT context. With respect to the use cases described in [I-D.seitz-ace-usecases] certificates may be short-lived and might need to contain attributes (which may be used for making access control decisions) rather than purely relying on the identity of users and their devices.

For the purpose of dynamic provisioning short-lived certificates, this document envisions to re-use a subset of the functionality offered by protocols like the Certificate Management over CMS (CMC) [RFC5272], the Certificate Management Protocol (CMP) [RFC4210], the Simple Certificate Enrollment Protocol [I-D.nourse-scep], Certification Request Syntax Standard - PKCS#10 [RFC2315] (with TLS or with PKCS#7 [RFC2986]). While these protocols offer slightly different features, on a high-level the all fulfill the same function. Note that the management of trust anchors may be provided by a different protocol, such as Trust Anchor Management Protocol (TAMP) [RFC5934].

Of course, certificates do not necessarily need to be short-lived and could even be provisioned during the manufacturing process and never changed during the lifetime of the device. The drawback of such an approach is, however, that mechanisms for certificate revocation have to be provided. Furthermore, privacy concerns might be arise since the same client certificate content will be shown to every service rather than information that is only relevant for a specific purpose.

```
                  +-------------+
                  |Certification|
                  |  Authority  |
                  +-------------+
                  ^        /
       Request   /        /
       (Short)  /        /
       Lived   /        /(Short) Lived
       Cert   /        / Certificate
             /        /
            /        /
           /        /
          /        v
        +-----------+                             +-----------+
        |           |   DTLS with certificate     |           |
        |           |   or app layer msg w/cert   |           |
        |  Client   |---------------------------->|  Service  |
        |           |<===========================>|           |
        +-----------+      Application Data       +-----------+
```

Figure 7: Certificate Model.

Pros:

- o Re-uses existing protocols: DTLS (or application protocol), CMP/ CMC/PKCS#10/SCEP, specifications (certificate format - RFC 6818 [RFC6818]), and concepts (PKI).

- o Large deployments on the Web and with enterprise system exist.

- o Products and open source code exists.

- o The certificate format offers flexibility in terms of content. New extensions have been defined over time.

- o Certificates can be used with DTLS without any additional modifications. Certificates can also used with application security mechanisms.

- o Authorization information may be placed in an extension of a public key certificate or in a separate attribute certificate [RFC3281]. Earlier work on KeyNote [RFC2704]could be re-used as it provides a more flexible authorization policy language.

- o A single certificate can be used with a number of different services.

- o Various PKI management protocols have been defined and they offer some flexibility. The properties vary on the specific use cases.

Cons:

- o The certificate format and the PKI management protocols use ASN.1.

- o No UDP or CoAP transport is defined for CMC/CMP/SCEP. For PKCS#10 no transport is defined at all.

- o The public key infrastructure only focuses on asymmetric cryptography. A separate body of work is available for provisioning symmetric keys (like one-time-keys), such as the Portable Symmetric Key Container (PSKC) [RFC6030] and Dynamic Symmetric Key Provisioning Protocol (DSKPP) ([RFC6063]).

- o Protocols for certificate enrollment are in use but many deployments use their own strategy for distributing certificates (typically long-lived) to their users.

- o Asymmetric cryptography is computationally more expensive than symmetric cryptography but offers additional security benefits.

3. Conclusion

Several existing protocols can be used to meet the use cases outlined in [I-D.seitz-ace-usecases]. Each technology presented here offers a number of possibilities for profiling to make them work on for constrained devices. Despite the range of available security protocols, the use cases suggest that there is a need to profile and to extend those in order to make them fit for the constrained environment.

The right choice of authentication and authorization protocol will heavily depend on the envisioned usage environment.

It is, however, also worth noting that several aspects that are not discussed in this document although they appear as requirements in the use case document, namely

- o a language for describing access control policies,
- o the encoding of these policies, and
- o the container for associating these policies with keying material.

4. Security Considerations

This entire document is about security.

5. IANA Considerations

This document does not require any actions by IANA.

6. Acknowledgements

The author would like to thank Stefanie Gerdes for her review comments.

7. Informative References

[I-D.ietf-abfab-arch]
Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", draft-ietf-abfab-arch-10 (work in progress), December 2013.

[I-D.ietf-kitten-sasl-oauth]
Mills, W., Showalter, T., and H. Tschofenig, "A set of SASL Mechanisms for OAuth", draft-ietf-kitten-sasl-oauth-12 (work in progress), December 2013.

[I-D.ietf-lwig-terminology]
Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained Node Networks", draft-ietf-lwig-terminology-06 (work in progress), December 2013.

[I-D.ietf-oauth-json-web-token]
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token-15 (work in progress), January 2014.

[I-D.ietf-oauth-v2-http-mac]
Richer, J., Mills, W., Tschofenig, H., and P. Hunt, "OAuth 2.0 Message Authentication Code (MAC) Tokens", draft-ietf-oauth-v2-http-mac-05 (work in progress), January 2014.

[I-D.nourse-scep]
Pritikin, M., Nourse, A., and J. Vilhuber, "Simple Certificate Enrollment Protocol", draft-nourse-scep-23 (work in progress), September 2011.

[I-D.seitz-ace-usecases]
Seitz, L., Gerdes, S., and G. Selander, "ACE use cases", draft-seitz-ace-usecases-00 (work in progress), February 2014.

[I-D.tschofenig-oauth-hotk]
Bradley, J., Hunt, P., Nadalin, A., and H. Tschofenig, "The OAuth 2.0 Authorization Framework: Holder-of-the-Key Token Usage", draft-tschofenig-oauth-hotk-03 (work in progress), January 2014.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

[RFC2704] Blaze, M., Feigenbaum, J., Ioannidis, J., and A. Keromytis, "The KeyNote Trust-Management System Version 2", RFC 2704, September 1999.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, November 2000.

[RFC3281] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, August 2008.

[RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, June 2008.

[RFC5934] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Management Protocol (TAMP)", RFC 5934, August 2010.

[RFC6030] Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, October 2010.

[RFC6063] Doherty, A., Pei, M., Machani, S., and M. Nystrom, "Dynamic Symmetric Key Provisioning Protocol (DSKPP)", RFC 6063, December 2010.

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.

[RFC6818] Yee, P., "Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 6818, January 2013.

Author’s Address

Hannes Tschofenig
ARM Ltd.
110 Fulbourn Rd
Cambridge CB1 9NJ
Great Britain

Email: Hannes.tschofenig@gmx.net
URI: http://www.tschofenig.priv.at