How much Security for Switching a Light Bulb – The SOA Way

Sebastian Unger
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Germany
Email: sebastian.unger@uni-rostock.de

Stefan Pfeiffer
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Germany
Email: stefan.pfeiffer@uni-rostock.de

Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Germany
Email: dirk.timmermann@uni-rostock.de

***Abstract*—What visions of technologies such as the Internet of Things (IoT), Pervasive Computing (PC) or Ambient Intelligence (AI) have in common is that they employ a very high amount of critically resource-constrained devices. This of course raises a whole set of new security challenges. Instead of proposing another middleware for IoT, PC or AI, and presenting a security approach for it we focus on existing, widely-adopted security mechanisms and concepts and give a comprehensive overview. Furthermore, we examine the Web Service security suite as an example for a comprehensive security framework on application level. We use an intentionally kept simple scenario including light bulbs and switches to demonstrate necessary security mechanisms and properties and to judge applicability of the presented existing security techniques and frameworks.**

Keywords—Devices Profile for Web Services, DPWS, Internet of Things, Pervasive Computing, Security, SOA for devices

I. Introduction

Visions such as the Internet of Things (IoT), Pervasive Computing (PC) or Ambient Intelligence (AI) – where every day objects communicate with each other to enrich and simplify our daily lives – raise a whole new set of security issues. This particularly results from the use of critically resource-constrained embedded computing devices. This in turn results from the requirement for these objects to be as energy-saving as possible since it appears inadequate for an IoT-enabled light switch to consume around 50 watts just so it can switch a light bulb securely. This becomes even more critical if one considers, that usually there is not only one switch but dozens, furthermore networking fridges, A/Cs, toys, door bells, HiFi-racks, humidity-, noise- or temperature sensors and the like. In a smart home this easily sums up to hundreds of computing devices and thousands or even ten thousands in smart office buildings. Returning to a switch and a light bulb, these devices form a very simple use case that allows us to focus solely on security issues.

A switch connected to a light bulb with ordinary copper wire provides its users with all the necessary security mechanisms. Authentication has been taken place at the time the devices have been wired with each other. Access control, also known as authorization, is an inherent feature of this setup, since unauthorized users have no access to the switch. Message integrity is completely ensured, since nobody can tamper with the command to turn a light bulb on or off. Eventually, confidentiality – if desired at all in this scenario – is also ensured as long as no one has access to the wire and tries to figure out if it carries any current (unless there are windows – but this is the same as somebody watching your monitor while you read your encryptedly sent e-mails). But how about the IoT-enabled equivalents? What is necessary to restore (or even improve) these security properties? And how can it be achieved?

The main contributions of this paper is to give an overview over existing, widely-adopted security mechanisms considering service-oriented architectures for resource-constrained devices. We identify problems when applying these existing mechanisms to resource-constrained devices and describe our work in progress to overcome these difficulties. The remainder of this paper is structured as follows. In section II we explore the briefly described use case and construct an IoT-enabled equivalent while focusing on security issues. Section III describes the state of the art in service-oriented architectures for resource-constrained devices and applicable, existing security mechanisms. In section IV, we identify problems with the existing solutions and describe our future work on how to secure a network of critically resource-constrained devices. We give an outlook and conclude this paper in section V.

II. Switching the Light Bulb

In this section, we explore the briefly described scenario in detail. In section II-A we present a possible IoT-enabled hardware setup to switch light bulbs and compare it to the classical approach (switches, bulbs and copper). We want to stress again, that it is not our intention to construct complex visions of possible IoT applications but to leverage a scenario as simple as possible so we can focus on security issues in sections II-B and IV.

A. Hardware Setup

The classical and the IoT-enabled approach are depicted in figures 1 and 2. The setup in fig. 1 is a very basic one.

978-1-4577-1379-8/12/$26.00

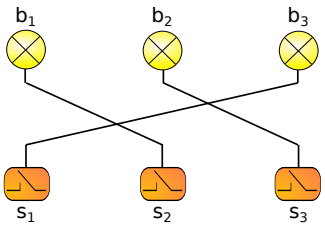

Fig. 1: Bulbs and Switches – the ordinary setup

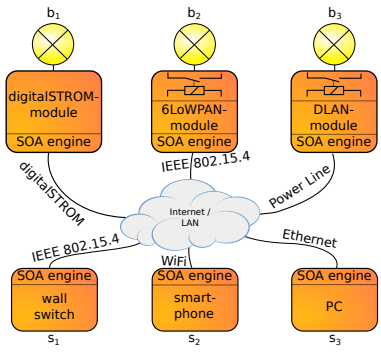

Fig. 2: Bulbs and Switches – an IoT-enabled setup

Three switches are connected to three bulbs with copper wire, so every switch is capable of switching a single light bulb. However, the IoT-enabled setup in fig. 2 is more sophisticated. As a switch one could imagine a device such as a regular wall switch. But instead of passively closing a circuit, it actively sends a command using a wireless connection such as 6LoWPAN (IPv6 over IEEE 802.15.4) ([1]). To eliminate the necessity of changing or recharging its batteries, energy harvesting technologies can be used. Of course, more powerful "switches" such as a smartphones, tablets or PCs can be considered, communicating over technologies such as WiFi, Ethernet, digitalSTROM ([2]) or power line communication (e.g. [3]). What all these technologies have in common is that they use the Internet Protocol (IP) so there is no need for proxies or gateways and if provided with SOA capabilities, they had a common semantic interface. The advantages of such a sophisticated setup are obvious. For example, installation – and more importantly changing existing installations – of switches and light bulbs becomes easier. Also, switches can be mobile so a carried smartphone automatically turns the light on and off in different rooms or hallways. A drawback is obvious as well. One needs some kind of infrastructure such as WiFi-Routers and IEEE 802.15.4 bridges – unless they are already present. To get an impression of the necessary security-related feature set, we will explore two very simple scenarios in the following section.

B. Security Issues

To figure out which security features are necessary, we have to declare the security-related behavior first. Thus, in this section we explore two scenarios and what the system has to ensure, what it has to provide and how it should behave.

1) Day-to-day Use: In this static scenario, we observe an existing installation of switches and light bulbs and how it should behave. We presume, that an initial security bootstrap routine has taken place earlier. This day-to-day-use scenario demonstrates the very basic security demands. If a switch sends a command to a light bulb to turn on or off, first *authenticity* and *integrity* of the message have to be verified. While the former identifies the device that sent the command (e.g. the light switch in the living room, not an arbitrary smartphone outside the building), the latter one ensures that the command was not altered by a third party while in transit (e.g. to forge an identity or to alter the command itself). After this, authorization of the command needs to be checked to figure out if the device is allowed to invoke this particular command (consider an apartment building where the janitor does the setup centralized - so knowing a switch does not include allowing it to turn on or off every possible light bulb). Eventually, confidentiality has to be ensured, so no illegitimate entity is able to eavesdrop the commands.

2) Replacing a light switch: Compared to the preceding scenario, the following is more dynamic. In this scenario, light switches are to be changed. However, the old ones are supposed to be reused (e.g. sold to a non-trustworthy third party). First, all authorizations concerning these switches have to be withdrawn so they no longer may send commands to light bulbs. This becomes an even more critical issue if connected homes are considered, where security information is delegated between different trust domains, so a notification must be sent to all connected realms. The second issue to consider is more subtle. It has to be ensured, that the switches that once were legitimate members of the network are not able to eavesdrop the encrypted traffic of the network they once were members of. Eventually, there is no need to delete any information regarding the authentication of the old switches – on the other hand, there is no reason to keep any of it.

Now that the old switches are removed securely, it is time to integrate the new ones. When they first come to life in their new home, some process takes place in which they announce their identity as a light switch and they discover light bulbs they could possibly turn on or off. A detailed solution for this problem is described briefly in section III-A. Besides this functional bootstrapping routine, there must happen some security bootstrapping to fulfill the following tasks. First, a new switch has to identify itself to the network so the latter one can authenticate the switches' commands later. To achieve this, a switch must gather information about possible ways to authenticate or at least announce how it plans to authenticate to the other participants. Eventually, some sort of cryptographic credentials are somehow exchanged, verified and stored. Along with authenticating new switches, the existing network has to grant authorizations, so a switch is actually allowed to turn on and off one or more light bulbs. Before being able to turn the lights on and off, a light switch again has to gather information about the cryptographic capabilities of the light bulbs it wants to switch, e.g. which cryptographic algorithms

it supports and which security policies it requires (for some light bulbs, confidentiality might be optional - so there is no need for en- and decryption).

3) Summary: Basing on the two scenarios, the necessary security requirements can be derived . The described system has to provide that every device is able to

- provide and verify message integrity
- provide and verify message authenticity
- en- and decrypt messages

Furthermore, there have to be mechanisms to

- securely remove devices from the existing network
- notify connected trust domains about devices that left the own domain
- seamlessly integrate new devices into the existing network (e.g. autonomously negotiate suitable authentication methods)
- exchange, verify and store cryptographic credentials
- let devices gather security-relevant information about each other
- share gathered information with connected trust domains

III. Related Work and State of the Art

During the past years, a high amount of research was dedicated to developing middlewares for the domains of IoT, PC or AI. These middlewares often also provide proposals for secure communication among embedded devices. As an outcome, there is a plethora of different, non-interoperable approaches on security frameworks for large distributed systems of interconnected embedded devices. For this reason, we want to recollect existing, widely adopted and well-accepted security techniques and investigate on their suitability for embedded devices. In this context, we distinguish between mechanisms provided by a network stack on the one hand and application layer security on the other hand. For the latter one, we chose the Web Service (WS) Security specification suite as an example because it comprises a security framework for large distributed systems. Furthermore, its base technology – Web Services – is already ported to the domain of embedded devices by means of the Devices Profile for Web Services (DPWS, [4]), which we briefly describe prior to examining the existing security techniques.

A. SOA for Devices: The Devices Profile for Web Services

Service oriented architectures (SOA) find their origins in the enterprise domain. Their basic idea is to loosely couple functional units using unified interfaces to avoid large monolithic software blocks that render impossible to be efficiently maintained when exceeding a certain size. The most popular implementation of the SOA concept is Web Services and its communication protocol SOAP. The basic concepts of SOAs – the loose coupling and the unified interface – caught the interest of embedded device developers. Initially set up by Microsoft, the Devices Profile for Web Services (DPWS) ([4]) was meant to become the successor of UPnP. However, in the meanwhile DPWS became a completely independent technology and provides resource-constrained devices such as printers with the capability to easily and seamlessly integrate into existing Web Service infrastructures in enterprises. While office supplies such as printers or scanners can be quite powerful in terms of computing resources, there are also movements to leverage DPWS in medical supplies ([5]), commercial home automation ([6]) or wireless sensor networks (WSN) ([7]). Even real-time approaches are considered ([8]).

In general, DPWS distinguishes between *DPWS Devices* and *DPWS Clients*. While the former ones may provide an arbitrary number of services, they can be invoked by the latter ones. DPWS provides Clients with the capabilities to dynamically discover Devices at run time without the necessity of a central registry and to fetch metadata from them that contain e.g. device type, model number or the formal interface representation in terms of a WSDL description. After these steps, services and their methods on Devices can be invoked by Clients. In addition to this traditional request-response-pattern, DPWS offers a publish-subscribe mechanisms by leveraging the specification WS-Eventing ([9]).

Regarding the scenario in section II-A, light bulbs can be considered as Devices, hosting a Service that provides methods for switching the light on and off, to retrieve the current status or to dim the light. The switches on the other hand are the Clients invoking these methods.

The security concept of DPWS consists of so-called security profiles. A profile can be understood as a set of protocols and rules two parties agree on before connecting to each other for the first time. The DPWS specification defines such a security profile as a security-related interoperability guideline. It is optional and may be replaced by any other security profile. This profile basically assumes that every device owns an X.509 certificate. Discovery messages (sent over UDP) are digitally signed using the certificate's PKI. The remaining traffic is secured using TLS (see section III-B). Due to its intentionally preserved simplicity this profile lacks of a certain set of features. For example, there is no explanation on how to provide necessary certificates which means that no authentication mechanism between devices is defined. It also lacks of a flexible authorization system. Eventually, it provides no way to use security mechanisms different to TLS such as embedding cipher texts into SOAP messages with XML-Encryption or use symmetric cryptography which is more suitable for critically resource-constrained devices. Eventually, a Device offers security (by providing an encrypted channel by TLS) and a hosted Service may decide to make use of this offer. There is no support for setting up fine-grained security configurations where e.g. every method of a service may define own security policies.

There are several approaches on alternative security profiles. For example [10] proposes a security profile suitable for few devices in an automotive context and [11] introduces a comprehensive security architecture for huge office networks to integrate printers and the like into an existing SOA infrastructure. When it comes to highly resource-constrained environments, in [12] the authors describe an approach which relies on offline bootstrapping and is highly vulnerable. But

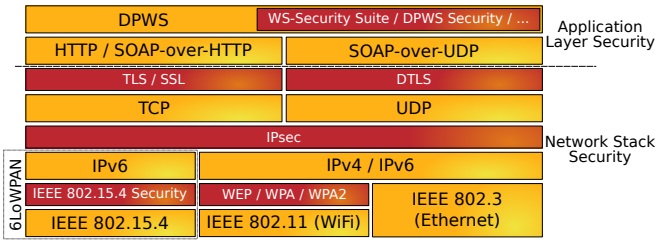

Fig. 3: Network stack and application layer security

a drawback that all these approaches have in common is that they match the problem statement formulated earlier. They describe a security solution that matches a special case but is unlikely to interoperate with other approaches.

B. Network Stack Security

As mentioned earlier, we want to recollect the suitability of widely-adopted and well-accepted security mechanisms to increase interoperability of different systems in the field of IoT, PC or AI. The first part of the covered techniques is summarized as network stack security since they usually reside within the network stack implementation of every common operating system or are provided as a library. As depicted in figure 3, MAC layer security, IPsec and (D)TLS are described briefly in the following.

On link layer, Ethernet (IEEE 802.3) and its wireless equivalent WiFi (IEEE 802.11) are widely spread and accepted for interconnecting computers and consumer electronics devices in a plethora of different domains. Both of them are designed with their focus on high throughput. When it comes to low power dissipation – often desired by mobile battery-driven resource-constrained devices such as sensors and simple actuators – IEEE 802.15.4 forms a promising approach which gets a lot of attention these days due to its employment in ZigBee and 6LoWPAN ([1]).

Due to its wired nature, Ethernet comes with an inherent security property, since an attacker needs physical access to the wire. To provide WiFi connections with an equivalent degree of security, there are several protocols. Since Wired Equivalent Privacy (WEP) is known to be unsecure ([13]), and Wireless Protected Access (WPA) is suspected to be as well ([14], [15]), its successor WPA2 is the recommended choice. The specification IEEE 802.15.4 considers security as well. Basically, it provides four modes of operation: No security at all, encryption only, providing message integrity and authenticity and the combination of the latter two. Providing and managing keys as well as any kind of authentication is out of the scope of this specification and is supposed to happen on application level.

However, security on link layer is only supposed to restore the same degree of security wired networks provide. Depending on the protocol and of the mode of operation, every legitimate member of a network can eavesdrop every transmitted packet or the maintenance effort for managing keys is very high. Link layer security also does not provide end-to-end-security between two endpoints when packets have to be routed, since its security mechanisms end at the border of the local network.

To close this gap, designers created the protocol suite IPsec when specifying IPv6, which is optionally also available for IPv4. IPsec consists of two independent protocols. *Authentication Header* provides authenticity and integrity, the *Encapsulated Security Payload* additionally provides confidentiality. Besides, IPsec provides two modes. The Transport Mode provides security between two machines. In Tunnel Mode a secure channel can be established between two local networks, a local network and a single machine or between two machines. Since IPsec secures traffic above Internet layer, so above IP, packets can be routed securely. FERGUSON and SCHNEIER stated in [16], that though IPsec appears to be the best way to secure IP traffic, it is overly complex. ECKERT agrees in [17] and derives, that due to its complexity, IPsec is very difficult to configure and to maintain. Besides, the complexity leads to solutions of different vendors often not being interoperable. She also criticizes that parts of the protocol management are to be handled by the application layer, which violates the principle of layer models.

An alternative to IPsec is the protocol Transport Layer Security (TLS) ([18]) (formerly known as SSL). It resides above transport layer and thus immediately below application layer. It provides mutual authentication of two parties using public key infrastructures and X.509 certificates though authentication of client, server or both is optional[1]. TLS also encrypts traffic and ensures its integrity. Since TLS is by far less complex than IPsec, it is spread more widely. However, TLS depends on TCP on transport layer. To provide an equally secure channel using connectionless protocols, the protocol Datagram Transport Layer Security (DTLS) was developed ([19]). In contrast to TLS, DTLS relies on UDP. Anyway, the dependence on a certain transport layer protocol is the main drawback of (D)TLS.

C. Application Layer Security

The techniques in the preceding section usually reside in an operating system's network stack and thus are applicable to most protocols on application layer. However, they all only solve single aspects while comprehensive approaches are provided on application level. At this level, we focus on a framework which exists for many years already and is widely adopted and accepted: The WS-Security Suite. With the emerging popularity of Web Services in the business domain, security issues shifted into the focus. Thus, a set of security specifications for Web Services - called the WS-Security-Suite - were released. Their functionality and interaction is briefly described here. The specification WS-Security ([20]) describes, how a SOAP message can be digitally signed (leveraging XML-Signature) and how cipher texts can be embedded using XML-Encryption. It also defines, how additional cryptographic credentials - so-called

[1]In practice, the server is authenticated by TLS while clients are authenticated on a higher level by different means.

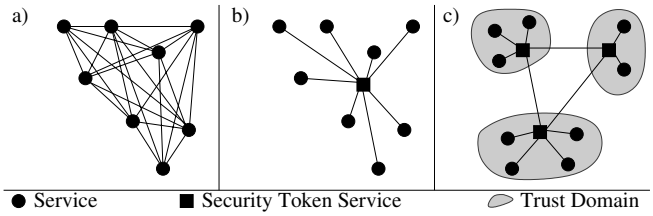

Fig. 4: Trust relationships
a) direct, b) indirect via STS, c) indirect among trust domains

security tokens, e.g. X.509-Certificates, username-password-combinations or Kerberos tickets - can be attached to a SOAP message. WS-SecurityPolicy ([21]) enables devices to announce their cryptographic capabilities and security policies to potential clients. While WS-Security provides mechanisms to secure a single SOAP message, WS-SecureConversation ([22]) provides mechanisms to establish a secure channel, similar to the mechanisms TLS uses. This way, performance can be improved by orders of magnitude. A secure context is represented by a secure-context-token. Such a token can be negotiated between two parties directly or it can be requested from a central trustworthy instance, a so-called Security Token Service (STS). The STS is defined by the specification WS-Trust ([23]). Its basic concept is to ease up authentication and credential exchange between members of a network, since trust relationships are always set up with the STS. This leads to implicit trust among all other parties (see figure 4a) and b)). An STS can also broker trust between different trust domains. If there is a trust relationship between STSs in different local networks, there is implicit trust between every participant of every network (fig. 4c)). While WS-Trust defines the messaging framework for trust brokering, the specification WS-Federation ([24]) defines concrete authentication protocols. A federation consists of several institutions willing to cooperate to achieve a common goal. Each participant provides services that are supposed to be used by other participants of the federation. The goal of WS-Federation is to share and manage security configurations such as trust and authorization information among the whole federation.

IV. Identified Problems and a solving Approach

In the preceding section, we presented security mechanisms and architectures for the Devices Profile for Web Services in particular (sct. III-A), for Web Services in general (sct. III-C) and mechanisms that are independent from any application (sct. III-B). Regarding network stack security, we showed that MAC layer security for wireless transmission merely restores the degree of security wired technologies inherently provide. More comprehensive security solutions are provided by IPsec and (D)TLS. However, the former is very complex and thus difficult to setup and maintain, while the latter one provides hop-to-hop encryption only and requires further efforts to safely cache or store encryptedly transmitted data. Besides, TLS requires the participating devices to cope with X.509 certificates and asymmetric cryptography. In addition to missing flexibility and missing authentication capabilities, this also is one of the main drawbacks of the DPWS security profile. Thus, the presented related work ([10], [11], [12]) proposes to use WS-Security mechanisms such as XML-Encryption instead or at least to provide this capability.

When Web Services got widely adopted in the enterprise domain, according security specifications were released. As described in section III-C, they form a comprehensive security framework for digital ecosystems comprising potentially thousands of participants whose simple functional units can be orchestrated to complex processes. The analogy to ambient intelligent systems is obvious when one considers e.g. a controller fetching temperature and humidity data, checking the house's inhabitants' calenders and thus opens or closes windows or adjusts the heating accordingly: several simple functional units form a complex system – here an automated room climate control. When it comes back to the scenario of light bulbs and switches and especially to the discussed security issues in section II-B, the WS-Security suite provides all necessary mechanisms. Within the day-to-day-use scenario, WS-Security ensures confidentiality and message integrity. By leveraging the security token profiles, additional authentication or authorization information can be attached. In addition, WS-SecureConversation allows two devices to establish a secure channel in which symmetric cryptography can be used which increases performance by orders of magnitude. In the second half of the scenario, where switches are to be replaced, WS-Trust offers opportunities to easily integrate the new switches into an existing infrastructure and can delegate trust-related and other cryptographic credentials between devices and even broker them between different trust domains by leveraging WS-Federation as well. The latter one also offers a vital feature called 'federated sign out'. When a device leaves a trust domain's responsibility (e.g. when selling light switches), all participating federation members get noticed about this and can withdraw all established trust relationships with this entity. Eventually, WS-SecurityPolicy and WS-Federation provide a comprehensive framework for clients to gather all kind of relevant information about devices in a network, such as supported cryptographic algorithms or required security properties.

This basically means that the WS-Security suite provides all necessary mechanisms to secure IoT-driven scenarios. However, these specifications were designed to run on desktop PCs and powerful web servers. Clients and servers are supposed to handle the following, resource-intensive tasks:

- *cryptography:* be capable of both symmetric and asymmetric encryption algorithms
- *authentication:* exchange and verify cryptographic credentials to establish trust relationships
- *authorization:* maintain information about legitimate consumers for every provided resource
- *connection management:* remember once established connections to increase performance (asymmetric vs. symmetric cryptography)
- *credential management:* store and maintain cryptographic

credentials

- *metadata and policy processing:* find, fetch and process information about other members of the network e.g. regarding supported algorithms or security policies
- *delegation between trust domains:* all the aforementioned tasks have to work across different trust domains (e.g. branch offices)

All of these tasks are very expensive, some in terms of memory (e.g. credential management) others in term of computational complexity (e.g. metadata and policy processing) which does not comply with the overall requirement on IoT-enabled devices to be small and energy-saving. However, we are convinced that visions such as the Internet of Things or Pervasive Computing will never become reality without comprehensive and yet discreet security concepts. Thus, the discussed security properties and related capabilities are inevitable.

Considering the very simple scenario with light bulbs and switches, this could lead to the question: *How costly (in terms of money or resources) are IoT-driven approaches when providing devices with the necessary security capabilities described above?* However, we consider another question more interesting: *Given a set of existing, widely adopted and long term well-proven security specifications for a similar technology – in terms of decentralization, loose coupling, orchestration and interconnection of different trust domains – that are not directly applicable due to the required resources: Is there a way to reuse and slightly alter them to benefit from their advantages?*

The answer to this question describes our generic approach and future work. Since we presume at least one powerful and trustworthy participant in every smart home or smart office, we will investigate on how many of the tasks described above can be offloaded from the resource-constrained clients and devices to the powerful and trustworthy participant within a trust domain. Some approaches are obvious such as letting the powerful participant fetch and process metadata and policies of two devices willing to communicate with each other and present a decision to them. Others could be more subtle e.g. offloading asymmetric cryptography to the powerful participant by raising authentication efforts or the total number of messages that have to be exchanged.

V. Conclusion and Outlook

In this paper, we define a scenario of an IoT-enabled setup of light bulbs and switches that we intentionally kept simple. We derive necessary security properties and give a comprehensive overview over existing, widely-adopted security mechanisms used by open Internet technologies and related work. Regarding security on application layer we focus on the Web Service Security suite and the Devices Profile for Web Services (DPWS) as an example for a service-oriented, distributed system.

We derive, that WS Security provides all necessary security features for complex setups of the domain of critically resource-constrained devices as they are found in IoT- or ambient-intelligence-scenarios and it is our future work to adapt the specification suite to the demands of resource-constrained embedded devices. This includes identifying atomic, security-related tasks and to investigate how to combine them in a secure manner, so gaining flexibility does not result in higher vulnerability. Based on this analysis, we will investigate on how much security-related functionality can be offloaded so we can reuse an existing, well-proven security framework.

References

[1] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Rfc 4944: Transmission of ipv6 packets over ieee 802.15.4 networks," Internet Engineering Task Force (IETF), Tech. Rep., September 2007.

[2] digitalSTROM.org, "digitalSTROM / Home," Online: http://www.digitalstrom.org/index.php?id=115&L=2, July 2011.

[3] HomePlug Powerline Alliance, "Home – HomePlug Powerline Alliance," Online: http://www.homeplug.org/home/, July 2011.

[4] OASIS, "Devices profile for web services version 1.1," July 2009.

[5] S. Pöhlsen, S. Schlichting, M. Strähle, F. Franz, and C. Werner, "A dpws-based architecture for medical device interoperability," in *World Congress on Medical Physics and Biomedical Engineering*, ser. IFMBE Proceedings. Springer Berlin Heidelberg, 2009, pp. 82–85.

[6] Exceptional Innovation, LLC, "Life|ware," Online: http://www.life-ware.com/technology/dpws.php, July 2011.

[7] G. Moritz, E. Zeeb, S. Prüter, F. Golatowski, D. Timmermann, and R. Stoll, "Devices profile for web services in wireless sensor networks: Adaptations and enhancements," in *IEEE Conference on Emerging Technologies Factory Automation*, September 2009.

[8] G. Moritz, S. Prüter, D. Timmermann, and F. Golatowski, "Web services on deeply embedded devices with real-time processing," in *IEEE International Conference on Emerging Technologies and Factory Automation*, September 2008, pp. 432–435.

[9] W3C: World Wide Web Consortium, "Web services eventing (ws-eventing)," March 2006.

[10] S. Unger, E. Zeeb, F. Golatowski, H. Grandy, and D. Timmermann, "Extending the devices profile for web services for secure mobile device communication," in *The 4th International Workshop on Trustworthy Internet of People, Things & Services in conjunction with Internet of Things Conference*, November 2010.

[11] J.-F. Martínez, M. López, V. Hernández, K. Jean-Marie, A.-B. García, L. López, C. Herrera, and C.-J. Sánchez-Alarcos, "A security architectural approach for DPWS-based devices," *CollECTeR Iberoamérica Conference*, 2008.

[12] V. Hernández, L. López, O. Prieto, J.-F. Martinéz, A.-B. García, and A. D. Silva, "Security framework for dpws compliant devices," in *Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies*, ser. SECURWARE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 87–92.

[13] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of rc4," in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, vol. 2259, pp. 1–24.

[14] T. Ohigashi and M. Morii, "A practical message falsification attack on wpa," 2009.

[15] E. Tews and M. Beck, "Practical attacks against wep and wpa," in *Proceedings of the second ACM conference on Wireless network security*, ser. WiSec '09. New York, NY, USA: ACM, 2009, pp. 79–86.

[16] N. Ferguson and B. Schneier, "A cryptographic evaluation of ipsec," Counterpane Internet Security, Inc, Tech. Rep., 2000.

[17] C. Eckert, *IT-Sicherheit, Konzepte - Verfahren - Protokolle*, 5th ed. Oldenbourg Wissenschaftsverlag GmbH, 2008.

[18] T. Dierks and C. Allen, "Rfc 2246: The tls protocol version 1.0," Internet Engineering Task Force (IETF), Tech. Rep., January 1999.

[19] E. Rescorla and N. Modadugu, "Rfc 4347: Datagram transport layer security," April 2006.

[20] OASIS, "Web services security: Soap message security 1.1," February 2006.

[21] ——, "Ws-securitypolicy 1.2," July 2007.

[22] ——, "Ws-secureconversation 1.3," March 2007.

[23] ——, "Ws-trust 1.3," November 2006.

[24] H. Lockhart *et al.*, "Ws-federation version 1.1," December 2006.