Title: Security and Key Establishment in IEEE 802.15.4

Student: Eirik Klevstad

Problem description:

Internet of Things (IoT) is a network where devices, sensors, vehicles, buildings, and humans communicate and collaborate, along with collecting and exchanging information. IEEE 802.15.4 specifies the lower layers for low-rate wireless networks, which are widely seen as the foundation for current IoT communications. One of the potential weaknesses of the IEEE 802.15.4 standard is the lack of specification for key establishment and management.

This thesis will focus on key management for device-to-device security in IoT. It will review and compare the proposed protocols, and include both formal and informal security analysis, as well as analysis of both key management requirements and key agreement protocol design for IoT security. Another goal of the thesis will be to suggest improvements and alternatives to the proposed protocols.

Responsible professor: Colin Boyd, ITEM

Supervisor: Britta Hale, ITEM

Abstract

This is my abstract.

Preface

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

List of Figures

List of Tables

List of Acronyms

6LoWPAN IPv6 over Low power Wireless Personal Area Networks.

AES Advanced Encryption Standard.

AES-CCM Advanced Encryption Standard Counter with CBC-MAC.

APKES Adaptable Pairwise Key Establishment Scheme.

BAN Burrows-Abadi-Needham.

CBC Cipher Block Chaining.

CCM Counter with CBC-MAC.

CIA Confidentiality-Integrity-Availability.

CTR Counter.

GPS Global Positioning System.

GUI Graphical User Interface.

IETF Internet Engineering Task Force.

IoT Internet of Things.

IP Internet Protocol.

MAC Media Access Control.

MSC Message Sequence Chart.

OSI Open Systems Interconnection.

RAM Random Access Memory.

RFID Radio Frequency Identification.

SNMP Simple Network Management Protocol.

SPDL Security Protocol Description Language.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

WPAN Wireless Personal Area Network.

WSN Wireless Sensor Network.

Chapter 1
Introduction

This should be the introduction to the thesis.

Chapter 2
Background

2.1 Internet of Things

Over the last decade, a concept called the *Internet of Things* has gained increased attention, both from the research community, as well as commercial actors and consumers. The term IoT is, accordingly to most sources, first introduced in 1999 by the British visionary Kevin Ashton in a presentation about Radio Frequency Identification (RFID) [3] [4]. Ashton's definition of the concept was a world where computers do not depend on human beings to provide them with information. Out of all the petabytes of information available on the Internet, most of it has been created and captured by humans performing some sort of action. In his opinion, IoT is more about providing computers with the ability to gather information on their own.

Another vision of the IoT is based upon the mixing of captured sensor data and connection of physical things to the Internet.

A computational device containing some sort of sensor is attached to your everyday physical device, creating a bridge between our physical world and the cyber world [14]. The connection to the Internet allows us to monitor and control these devices and sensors from a remote distance. Another vital part of IoT is device-to-device communications, essentially enabling devices to communicate with each other without human aid, and exchange and retrieve information.

The next step of IoT is device-to-device communication, efficiently enabling devices to retrieve and exchange information. Such devices could be sensors monitoring some operation, a physical area, or even devices attached to a physical body. The possibilities are more or less unlimited. Imagine a home automation and surveillance system for your cabin, where both lights, heaters, smoke detectors, underfloor heating, motion detectors, security cameras, garage and so on, are all interconnected with each other through small wireless devices. As it is called the *internet* of things for a reason, your system and devices would be accessible over the Internet, allowing you

to monitor the current status of your cabin remotely from your couch at home, as well as looking at historical data of the different sensors and devices. When the weekend arrives and you head for the mountains, the IoT provides you with an opportunity to preheat different (or all) sections of the cabin, deactivate the alarm, and perhaps instruct the sauna to start getting cosy.

Another approach is to avoid using a screen to remotely control the system, and instead allowing the system to observe and act on your behaviour. We want the devices to know us and figure out the correct thing to do without us telling them. For example, when pulling your car in the driveway, you want the garage door that is connected with your car to open up. The garage notifies your front door that you are home, which conveniently unlocks and notifies your house to turn on the lights in your hallway and perhaps the heater in your living room.

The possibilities that are revealed as the IoT grows larger and the services expands are infinite. The concept is highly applicable for different scenarios involving home automation, standalone consumer products, industrial and environmental facilities, as well as medical surveillance. While larger automation systems for homes and facilities have been the target for the research community and early adopters, the consumer market has been focused on so-called *wearables*. Wearables are fundamentally devices that you wear, such as smart watches, fitness trackers, virtual reality glasses, headphones and smart clothing. Such human-centric devices is less about automation, and more focused on personal improvement.

one can expect that the interest for human-centric scenarios of IoT will emerge in the years to come.

Nevertheless, the increase in IoT devices possibly provides us with a more cost efficient future, both in our use of time, as well as energy and consumption of other resources.

As the IoT is built upon the Internet, it faces the same types of security issues as the Internet itself. The amount of "things" connected to the Internet is calculated to be 6.4 billions by the end of 2016, which is almost a 30% increase from 2015. By 2020, the expected number of these "things" is more than 20 billions [2], providing attackers with equally many possible devices to attack. Given the knowledge that some of these devices may be medical (and other sensitive applications), we quickly recognize potential catastrophic scenarios.

The IoT architecture is divided into three layers; perception, transportation and application, each providing different types of security [13].Transportation and application layers, however, are out of scope for this project. Perception layer is mostly about sensors and other nodes that collects information from its environment,

and communicates it throughout the transport network. Another object for the layer is to pass control messages received. Examples of such technologies are RFID, Wireless Sensor Network (WSN) and Global Positioning System (GPS), each dealing with their own security problems and solutions. The most adjacent problems to the scope of this thesis are the problems related to key distribution and key management, which defines how two devices safely can establish secure communication between each other.

In an IoT world, the protection of data and privacy is an essential part. As previously mentioned, IoT technology may be a solution for problems involving sensitive information. In a medical facility, a possible scenario could be a WSN, which are dynamic and bi-directional network of nodes where each node has one or more sensors connected to it. A patient may have sensors implanted in its body, as well as different instruments attached for measuring different properties. All these devices communicate with each other wirelessly, and the network is therefore a possible target for an attacker. To prevent the attacker from eavesdropping, and possible forging content in the network, encryption and authentication of the different nodes are crucial.

Rewrite??

More motivation for the need for key management in IoT?

2.2 The IEEE 802.15.4 Standard

Following the evolution of IoT, the need for cheap devices to communicate efficiently between each other has arose. Existing architectures such as 802.11 (WiFi) and Bluetooth are too expensive in terms of processing and energy consumption, as the idea of IoT is to connect even the smallest devices to a network or Internet. As these devices are small, they have a limited amount of battery, and hence need to use it in a highly efficient matter.

802.15.4 protocols are envisioned for applications supporting smart homes, medical surveillance, monitoring systems for environmental and industrial systems, as well as sensor systems for heating and ventilation. As we know from the IoT, it is really the imagination that puts an end to the possibilities for interconnected devices. The OSI stack defines the internal structure of communications systems, and is shown in Figure 2.1.

As the 802.15.4 standard only defines the physical and data link (Media Access Control (MAC)) layer of the OSI stack, specifications need to be developed to utilize the possibilities provided by 802.15.4 in the upper layers. ZigBee, maintained by the ZigBee Alliance, is the most notable example of specifications that uses 802.15.4

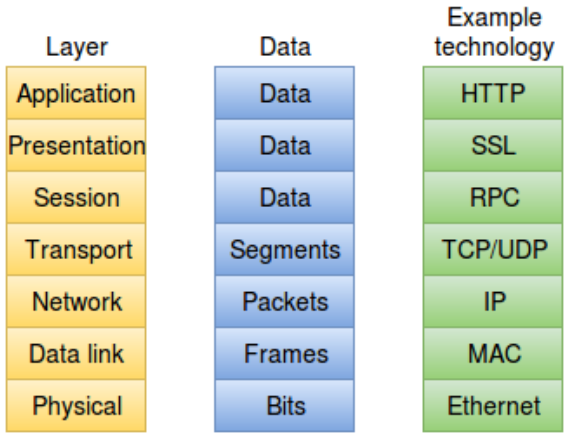

Figure 2.1: The OSI stack with layers, the data they carry, and some of the most known technologies for the different layers..

as its base. Others include WirelessHART, MiWi, and ISA100.11a. Cite these? Homepage?

The fundamental intention of the 802.15.4 standard is to provide low-rate, low-power communication between devices within a sensor network or Wireless Personal Area Network (WPAN). Its main use case is to let multiple devices within a short range communicate with each other over a low-rate radio, while maintaining a modest energy consumption. Figure 2.2 paints a picture of what 802.15.4 is compared to more well-established concepts such as WiFi (802.11) and Bluetooth, focusing on energy consumption, complexity and date rate. While being a smaller, more cost efficient device than those found in more complex networks, 802.15.4 networks have a much more limited range (about 10 meters), and in most cases a throughput below 250 Kbps [12]. Not only is the 802.15.4 standard significantly lighter in terms of data rate and power consumption, it is also aimed on a different market than regular WPANs. WPANs are oriented around a person, creating a personal network for the user, which has higher demands to data rate, and can allow a higher energy consumption. 802.15.4, however, focuses on interconnecting devices that do not necessary have this constraint, such as industrial and medical applications.

From a security point of view, the 802.15.4 standard provides two of the properties in the Confidentiality-Integrity-Availability (CIA) triad, namely confidentiality and integrity, through its link-layer security package [18]. When seen in conjunction with possible use cases for 802.15.4, these are important features. For sensor networks related to medical facilities, data confidentiality protects sensitive information about patients from being disclosed to unauthorized parties. Data integrity protects the

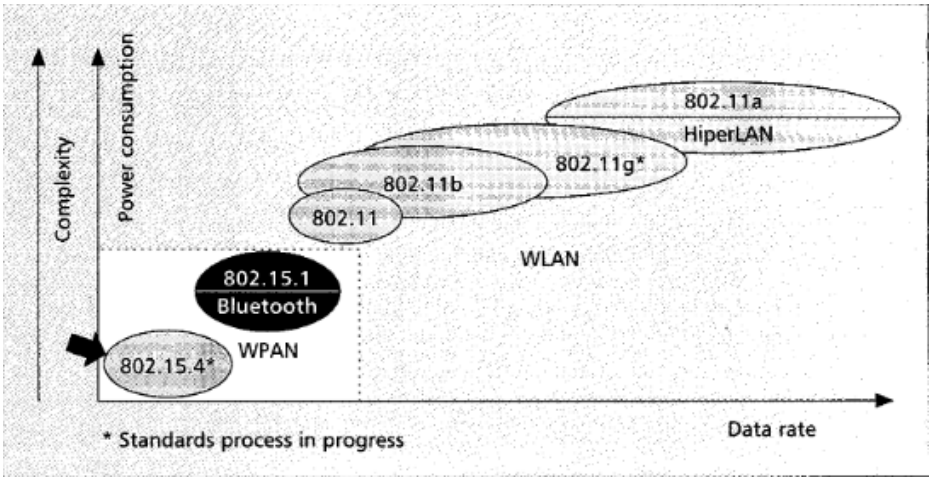

Figure 2.2: Figure of IEEE 802.15.4's operational space compared to other wireless standards [12]. REPLACE THIS

data from being tampered with, which can cause serious harm for all sorts of sensor networks and applications.

Four basic security services are provided in the 802.15.4 link-layer security package, namely access control, message integrity, message confidentiality and replay protection (sequential freshness). 802.15.4 is delivered with a total of eight different security suites, providing none, some or all of the described security services. In the most secure end of the scale we find Advanced Encryption Standard (AES)-Counter with CBC-MAC (CCM), which is encryption with either 32, 64 or 128-bit Message Authentication Code (MAC).

Access control and replay protection filters out packages at the link layer.

Message Integrity

Message confidentiality

2.3 6LoWPAN: Putting IP on Top of 802.15.4

Initially, the Internet Protocol (IP) was considered to be too "heavy" for low-power wireless networks such as the ones described by the 802.15.4 standard. The idea of implementing IP on top of 802.15.4 networks was born as early as in 2001 under the question "Why invent a new protocol when we already have IP?"[16]. With IP, the community already had a bundle of existing protocols for management, transport,

configuring and debugging, such as Simple Network Management Protocol (SNMP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), as well as standardized services for higher layers such as caching, firewalls, load balancing and mobility. Nevertheless, the initial idea of using IP in combination with sensor networks or WPANs was not accepted by various groups such as ZigBee [16]. The rejection did, however, not stop the initiative, and a group of engineers within Internet Engineering Task Force (IETF) started designing and developing what would later be known as IPv6 over Low power Wireless Personal Area Networks (6LoWPAN).

Another significant advantage with combining IP and 802.15.4 is the simplification of the connectivity model between various devices in the networks. As most 802.15.4-based specifications usually needs custom hardware that tend to be complex, the possibilities to interconnect different networks with each other is somewhat limited. By turning to IP, the need for complex connectivity models is obviate as it is possible to use well-understood technologies such as bridges and routers. Another advantage with using IP is that the routers between the 6LoWPAN devices and the outside networks (so-called edge routers) do not need to maintain any state as they are only forwarding datagrams.

The implementation of 6LoWPAN proved to not be any more expensive compared to other alternatives such as ZigBee or Z-wave in terms of code size and Random Access Memory (RAM) requirements [16]. Another important feature of 6LoWPAN

New features - Why is this doable over IP?

One of the main reasons for IP actually a feasible option for low-power wireless networks is its improved header compression algorithm. Its, at the time, unique concept is that you only "pay" for what you use in terms of overhead and processing. 6LoWPAN introduces a total of possible 256 different headers, each of them containing different fields based on its usage, and hence different lengths.

Encapsulation and header compression.

Supports common topologies such as star and mesh.

6LoWPAN uses, as stated in its acronym, IPv6.

Unique concept: Only pay for what you use in terms of overhead and processing because of a new header compressing mechanism.

Security considerations.

However, just like the IEEE 802.15.4 standard does not provide any description on how to deal with key management and key exchange, neither does 6LoWPAN.

2.4 Formal Security Analysis

As security protocols grows larger and more complex, they become more and more difficult for humans to analyse. One of the examples of the need for formal security analysis, is the Needham-Schroeder protocol [17] from 1978. The Needham-Schroeder protocol is based on public-key cryptography and was intended to allow two communicating parties to mutual authenticate each other.

Initially, the security protocol analysis was conducted using so-called Burrows-Abadi-Needham (BAN) logic, which showed promising results in finding security flaws and drawbacks for several authentication protocols, along with proving that the Needham-Schroeder protcol was, in fact, secure [7]. BAN logic is a set of rules which can be used to determine whether received information is legit or not by formally describe the interaction between communicating parties.

17 years later, after being deployed and widely used, G. Lowe discovered, using the automatic tool Casper, that the Needham-Schroeder protocol was in fact insecure, and vulnerable to a man-in-the-middle attack [15]. Formal security analysis gained increased interest from the research community after the discovery by Lowe, and ... More....

The Dolev-Yao model is a formal model used to prove the security properties of cryptographic protocols. While initially being a verification model built for public key protocols, the Dolev-Yao model is also the basis for most of the security analysis done by verification tools [8]. The model is built upon three primary assumptions [11]. Firstly, the Dolev-Yao model assumes that the cryptography is perfect, essentially meaning that the cryptographic system can not be tampered with, and an encrypted message can only be decrypted by the party possessing the corresponding decryption key. The second assumption is that the adversary has completely control over the communication network, hence he is able to observe all messages that are sent between communicating parties, and can inject messages given that he is able to forge its content in a valid matter. Lastly, messages that are sent in the network are to be observed as abstract terms, where the attacker has two possible outcomes; either he learns the complete content of the message, or he learns nothing at all.

More to come.

Chapter 3
Formal Security Analysis Using Scyther

There exists multiple state-of-the-art tools for performing formal analysis of security protocols, for example Avispa [1], ProVerif [5] and Scyther [6]. This thesis uses Scyther as the tool for conducting the formal security analysis, and the following chapter will give an introduction to Scyther, how it works, and examples of usages.

3.1 The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols

Scyther is a tool for both verification, falsification, and analysis of security protocols developed by Cas Cremers. The tool is based on a patter refinement algorithm that enables unbounded verification, falsification and characterization [9]. Scyther allows its users to verify security protocols in two different ways. The first option is to execute Scyther scripts through the command-line interface, which provides a pdf-file containing the results of protocol verification. Option two, is to use Scyther's own Graphical User Interface (GUI), which provides panels for both verification results, and in case of attacks being found; a visual graph of Scyther's proposed attack on the protocol. The most recent release of Scyther was published on April 4, 2014, and is currently available for both Windows, OS X and Linux.

Scyther performs its analyses under the *perfect cryptography assumption* [10]. This assumption states that all cryptographic functions are perfectly designed, hence it is impossible for the adversary to learn anything about the encrypted message without possessing the corresponding decryption key.

A security protocol's specification describes messages that are sent between parties and computation done at either party's side. This can be seen as the blueprint of what the different entities are actually allowed to do, and how

3.1.1 Scyther and its Pattern Refinement Algorithm / its novelty?

Most algorithms used in verification tools performs *bounded* verification, which means that they are capable of handling a finite space of protocol behaviours. This is a restriction, implying that the algorithm is not able to verify all possible behaviours (or states) of the protocol, but rather a finite subset of the possible state-space. One of Scyther's unique features is its ability to handle an *unbounded* (or infinite) state-space, and that it is guaranteed to *terminate* [10]. By constructing an algorithm that terminates every time, it allows for providing useful results, even when no attack is found, or if the algorithm is not able to verify the protocol in the unbounded state-space [9].

Scyther utilizes formal models for describing protocol behaviour. As these are symbolic, they are

As mentioned, a protocol specification contains a set of roles which serves as blueprints that describes what the protocol is able to do. An execution of the protocol is referred to as a *run*, and defines an unique instance of the protocol with respect to local constraints and the binding between the role and the actual agent acting out the roles behaviour.

Other novel features implemented in Scyther includes assisted protocol analysis by providing the analyst with different classes of protocol behaviours (or attacks), as well as so-called multi-protocol analysis which allows for analysing multiple protocols that coexist in the environment.

In the event of Scyther finding an attack on the modelled protocol, Scyther will also provide proves for the attack and present a graphical view of its proposed attack.

Provides formal proofs of the security protocol in question. If Scyther finds a possible attack, it will also prove it.

For analysis of protocol security, Scyther provides different classes of attacks, which deflects from other available analysis tools.

3.1.2 Reasons for Choosing Scyther

"Scyther is chosen because of reasons."

Another reason for choosing Scyther is that it is a teaching tool, and comes with a manual and a set of exercises for aiding the user in understanding the tool and its functionality.

Move this to the end of chapter?

3.2 Scyther's Different Modes of Usage

3.2.1 Verifying Claims

When modelling security protocols in Scyther, an important part is to state security claims for each of the different parties (or *roles*) of the protocol. Examples on such claims are that the key used for encryption between two roles is supposed to be secret, or that the communicating roles have authenticated each other correctly during the protocol execution.

```
claim(Alice, Secret, AliceSecretKey);
```

3.2.2 Generating Claims Automatically

If we were to develop a security protocol and not knowing the proper claims that should go with it, Scyther is able to automatically analyse our protocol and generate the appropriate claims. Scyther does so by claiming that all locally generated values and variables (such as cryptographic nonces) are supposed to be secret, before verifying them as described in the previous section.

3.2.3 Complete Characterization

In addition to verifying claims described in a protocol, Scyther is also able to present a finite set of traces that represent all the execution paths of the protocol. This information can be used to gain insight in potential problems and weaknesses of the modelled protocol. For most protocols, the set of such traces are limited to 1-5 paths, making it fairly easy to quickly obtain an oversight of the protocol's behaviour [9].

3.3 Claims

3.3.1 Alive

3.3.2 Secret

3.3.3 Non-injective Synchronization

Synchronization requires that all protocol messages occur in the expected order with expected values, and that the behaviour is equivalent to as if the protocol was executed without the presence of any adversary. An injective synchronization property states that the protocol executes as expected over *multiple* runs, claiming that it is not possible for an attacker to use information from previous runs into

disrupting the current protocol execution [8]. Such an attack is known as a replay attack, and is used by an adversary to inject traffic into the protocol execution to induce undesirable or unexpected behaviour.

Scyther, however, does not support this enhanced form of synchronization, hence it is vulnerable to replay attacks [9]. Using its language, we can claim that a property holds non-injective synchronization by using the `ni-synch` statement.

3.3.4 Non-injective Agreement

(weak agreement)

3.3.5 Running, Commit

3.4 Scyther Syntax

The syntax used in `.spdl`-files, which are protocol files that can be run and verified by Scyther, can resemble popular object-oriented languages such as C, C++ or Java. The structure of a minimum working example for the protocol known as Adaptable Pairwise Key Establishment Scheme (APKES) is shown below, consisting of an outer class defining the protocol and multiple agents (or roles) inside the protocol. In this example, we define that our protocol consists of two communicating parties; U and V, without giving them any specific behaviour.

```
protocol APKES(U, V){
  role U { };
  role V { };
};
```

For each of the different roles in the protocol, behaviour can be added as a sequence of send and receive events, as well as variable declarations, constants and claims. For our Alice role, we can define a simple behaviour as shown below, where Alice generates a random nonce `Na` and sends it to Bob, before receiving a message from Bob containing the random nonce `Nb`. All events are labelled with either `send` or `recv` followed by a subscript and a number. The number indicates the message's position in a Message Sequence Chart (MSC), and must be incremented for each message. Typically, a `send`-event has a corresponding `recv`-event at the receiving role with the same number.

```
role U{
  fresh Ru: Nonce; # A freshly generated nonce
  var Rv: Nonce; # A variable for receiving a nonce
```

```
  send_1(U, V, Ru); # Message sent from U to V containing Ru
  recv_2(V, U, Ru, Rv); # Message sent from V to U
    containing Ru and Rv, which is stored as a variable
};
```

Along with support for creating fresh nonce, variables and terms, Scyther also provides a wide set of cryptographic elements such as hash functions, symmetric-key cryptography, public-key cryptography, as well as declaring user specific types and macros, which are abbreviations of complex expressions into simpler once. A sequence of events within a role are in most cases concluded with a set of claim events. Claim events are, as previous mentioned, used for describing the security properties of the role. For our Alice, we want to claim that the two nonces `Na` and `Nb` are meant to be kept secret from an adversary.

```
role Alice{
  [...]

  # Claims:
  claim(Alice, Secret, Na);
  claim(Alice, Secret, Nb);
};
```

Probably elaborate more on the Scyther syntax? More examples?

3.5 Scyther's Graphical Interface for Verification and Falsification

When running Scyther in its three modes, different views are provided through Scyther's GUI. If we continue on our example from the section on Scyther's syntax, we now want to verify the security claims for Alice's role in the protocol.

Chapter 4
6LoWPAN Security - Adding Compromise Resilience to the 802.15.4 Security Sublayer

4.1 Ideas of the Paper

APKES. Threeway handshake.

4.2 Informal Analysis

...

4.3 Formal Analysis

4.3.1 Scyther

Script. Explanations. Compare.

Chapter 5
Krentz Mobility Paper

AKES. Allow nodes to leave and join network. Discover new neighbours. Reboot.

Chapter 6
TBA

Chapter 7
Conclusion

Chapter 8
Appendix

References

[1] Avispa. http://www.avispa-project.org/. Accessed: 2016-02-23.

[2] Gartner: Internet of Things. http://www.gartner.com/newsroom/id/3165317. Accessed: 2016-03-31.

[3] Internet of Things Phrase. http://www.rfidjournal.com/articles/view?4986. Accessed: 2016-03-31.

[4] Internet of Things Phrase. www.theguardian.com/sustainable-business/2015/feb/27/the-internet-of-things-what-the-man-who-coined-the-phrase-has-to-say. Accessed: 2016-03-31.

[5] ProVerif. http://prosecco.gforge.inria.fr/personal/bblanche/proverif/. Accessed: 2016-02-23.

[6] Scyther. https://www.cs.ox.ac.uk/people/cas.cremers/scyther/index.html. Accessed: 2016-02-23.

[7] Burrows, M., M. Abadi, and R. M. Needham (1989). A Logic of Authentication. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, Volume 426, pp. 233–271. The Royal Society.

[8] Cremers, C. and S. Mauw (2005). Operational Semantics of Security Protocols. In *Scenarios: Models, Transformations and Tools*, pp. 66–89. Springer.

[9] Cremers, C. J. (2008a). The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Computer aided verification*, pp. 414–418. Springer.

[10] Cremers, C. J. (2008b). Unbounded Verification, Falsification, and Characterization of Security Protocols by Pattern Refinement. In *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 119–128. ACM.

[11] Dolev, D. and A. C. Yao (1983). On the Security of Public Key Protocols. *Information Theory, IEEE Transactions on 29*(2), 198–208.

[12] Gutierrez, J. A., M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile (2001). IEEE 802.15.4: A Developing Standard for Low-power Low-cost Wireless Personal Area Networks. *network, IEEE 15*(5), 12–19.

[13] Jing, Q., A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu (2014). Security of the Internet of Things: Perspectives and Challenges. *Wireless Networks 20*(8), 2481–2501.

[14] Kopetz, H. (2011). *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Chapter Internet of Things, pp. 307–323. Springer.

[15] Lowe, G. (1996). Breaking and Fixing the Needham-Schroeder Public-key Protocol Using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 147–166. Springer.

[16] Mulligan, G. (2007). The 6LoWPAN Architecture. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, pp. 78–82. ACM.

[17] Needham, R. M. and M. D. Schroeder (1978). Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM 21*(12).

[18] Sastry, N. and D. Wagner (2004). Security Considerations for IEEE 802.15.4 Networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, pp. 32–42. ACM.