Department of Computer Science

Security in Emerging Networking Technologies

Views on Internet of Things and Software-Defined Networking

Markku Antikainen

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 275/2016

Security in Emerging Networking Technologies

Views on Internet of Things and Software-Defined Networking

Markku Antikainen

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall H304 of the school on 4th January 2017 at 12 noon.

Aalto University
School of Science
Department of Computer Science

Supervising professor
Professor Tuomas Aura

Preliminary examiners
Associate Professor Panos Papadimitratos, KTH Royal Institute of Technology
Associate Professor Stefan Schmid, Aalborg University

Opponent
Professor Olaf Maennel, Tallinn University of Technology

Aalto University publication series
DOCTORAL DISSERTATIONS 275/2016

© Markku Antikainen

ISBN 978-952-60-7218-0 (printed)
ISBN 978-952-60-7217-3 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-7217-3

Unigrafia Oy
Helsinki 2016

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Markku Antikainen

Name of the doctoral dissertation
Security in Emerging Networking Technologies
Views on Internet of Things and Software-Defined Networking

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 275/2016

Field of research Computer Science

Manuscript submitted 19 August 2016 **Date of the defence** 4 January 2017

Permission to publish granted (date) 18 October 2016 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Internet of Things (IoT) and software-defined networking (SDN) are two recent trends that are believed to dramatically shape the future computer networking. IoT connects ubiquitous devices that will fill our surroundings and help people in everyday tasks, while SDN reshapes the way computer networks are managed. These developments will also change the information security and threat landscape of the networks. The IoT devices have often constrained user interfaces, which creates challenges for security configuration and protocols. Software-defined networking, on the other hand, changes the network architecture and thus exposes the infrastructure to new threats. Identifying potential security problems and finding solutions for them is paramount for the success of these technologies.

This thesis studies a set of problems relating to the security of emerging networking technologies and, more specifically, IoT and SDN. We describe a novel device pairing protocol for IoT devices that uses fuzzy user-generated information for authenticating the key exchange. Unlike earlier proposals, the protocol does not make any assumptions on how entropy is distributed in the shared secret. This makes the protocol well-suited for situations where the shared secret is fuzzy and its properties are variable. We also study SDN security from various angles. More specifically, we discuss insider attacks where the adversary has already gotten a foothold in the target network, and denial-of-service attacks that can be launched via the network data plane. We find that the consequences of these attacks are highly dependent on the network configuration. The thesis also describes a tenant isolation solution implemented with existing SDN technologies. Finally, we analyze the security of Bloom-filter-based authorization mechanisms for multicast forwarding and find that several of the proposed protocols are vulnerable to denial-of-service attacks and are thus unsuitable for open networks.

Keywords computer security, Internet of things (IoT), software-defined networking

ISBN (printed) 978-952-60-7218-0 **ISBN (pdf)** 978-952-60-7217-3

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2016

Pages 136 **urn** http://urn.fi/URN:ISBN:978-952-60-7217-3

Tekijä
Markku Antikainen

Väitöskirjan nimi
Tietoturvallisuus tulevaisuuden tietoverkkoteknologioissa
Näkökulmia esineiden Internetiin ja ohjelmoitaviin tietoverkkoihin

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietotekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 275/2016

Tutkimusala Tietotekniikka

Käsikirjoituksen pvm 19.08.2016 **Väitöspäivä** 04.01.2017

Julkaisuluvan myöntämispäivä 18.10.2016 **Kieli** Englanti

☐ **Monografia** ☒ **Artikkeliväitöskirja** ☐ **Esseeväitöskirja**

Tiivistelmä

Esineiden Internet (Internet of Things) ja ohjelmoitavat tietoverkot (software-defined networks) tuovat merkittäviä muutoksia tietoverkkojen toimintaan. Esineiden Internet yhdistää jokapäiväisen ympäristömme tietotekniset laitteet, ja ohjelmoitavat tietoverkot muuttavat tapaa jolla verkkoja hallitaan. Näistä muutoksista seuraa uusia tietoturvahaasteita ja uhkia. Jokapaikan tietotekniikka luo haasteita laitteiden konfiguroinnille ja tietoturvaprotokollien suunnittelulle, koska laitteissa on hyvin rajalliset käyttöliittymät. Ohjelmoitavat tietoverkot taas muuttavat verkkoarkkitehtuuria radikaalisti, mikä avaa uusia mahdollisuuksia hyökkääjille. Tietoturvauhkien tunnistaminen näissä järjestelmissä ja ratkaisujen kehittäminen niihin on välttämätöntä, jotta uudet teknologiat voidaan ottaa käyttöön.

Tämä väitöskirja käsittelee tulevaisuuden tietoverkkoteknologioiden, erityisesti esineiden Internetin ja ohjelmoitavien tietoverkkojen, tietoturvallisuutta. Väitöskirjassa esitetään avaimenvaihtoprotokolla käyttöliittymältään rajoitettujen laitteiden paritukseen hyödyntämällä käyttäjän tuottamaa sumeaa salaisuutta. Esitetty protokolla tekee vähemmän oletuksia jaetun salaisuuden entropiajakaumasta kuin aiemmat vastaavat menetelmät, joten se sopii tilanteisiin, joissa sumean salaisuuden ominaisuudet vaihtelevat. Ohjelmoitavien tietoverkkojen turvallisuutta väitöskirja käsittelee useista eri näkökulmista. Tutkimuksessa analysoidaan sisäpiirihyökkäyksiä, joissa osa verkkolaitteista on hyökkääjän hallinnassa, sekä verkon data-kerroksen kautta tehtäviä palvelunestohyökkäyksiä. Analysoitujen uhkien vakavuus riippuu paljon tietoverkon rakenteesta ja asetuksista. Väitöskirjassa esitetään myös mekanismi asiakkaiden eristämiseen toisistaan jaetussa tietoverkossa. Lisäksi väitöskirjassa analysoidaan tilattomiin ryhmälähetysprotokolliin ehdotettuja ratkaisuja palvelunesto-ongelmien estämiseksi ja osoitetaan ne riittämättömiksi, mikä rajoittaa kyseisten protokollien käyttöä avoimissa tietoverkoissa.

Avainsanat tietoturvallisuus, esineiden internet (IoT), ohjelmoitavat tietoverkot (SDN)

ISBN (painettu) 978-952-60-7218-0 **ISBN (pdf)** 978-952-60-7217-3

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Helsinki **Painopaikka** Helsinki **Vuosi** 2016

Sivumäärä 136 **urn** http://urn.fi/URN:ISBN:978-952-60-7217-3

Preface

This book is a result of my studies done in Aalto University's Department of Computer Science between 2013 and 2016.

I am grateful to Tuomas Aura for his guidance, support, and patience. Thanks to co-authors: Mikko Särelä, Mohit Sethi, Rajat Kandoi, Sinisa Matetic, and Alireza Ranzibar. My research visit to Cambridge made my grad-school experience more complete. I would like to thank Arjuna Sathiaseelan, Liang Wang et al. for making this visit possible and, more importantly, fun. Thanks to my office-mates Sandeep Tamrakar, Thanh Bui, Vilen Looga and countless of other colleagues with who I have shared the grad-student's agony. I would also like to thank the pre-examiners Stefan Schmid and Panos Papdimitratos whose comments helped me to improve the quality of the thesis.

The work was funded by the Helsinki Doctoral Education Network in Information and Communications Technology (HICT) and by TEKES as a part of the IoT and Cyber Trust programs of DIGILE (the Finnish Strategic Center for Science, Technology and Innovation in the field of ICT and digital business). My research visit to the University of Cambridge was further supported by the Foundation for Aalto University Science and Technology, Kordelin Foundation, and HICT. Additionally, Nokia Foundation, and Finnish Foundation for Technology Promotion (Tekniikan edistämissäätiö) have generously supported my scientific endeavor.

My family and friends, Elina in particular, made this work more enjoyable.

Helsinki, December 11, 2016,

Markku Antikainen

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Sethi M., Antikainen M., Aura T. Commitment-based device pairing with synchronized drawing. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 181–189, 2014.

II Antikainen M., Sethi M., Matetic S., Aura T. Commitment-based device-pairing protocol with synchronized drawings and comparison metrics. *Pervasive and Mobile Computing*, vol. 16, part B, pp. 205–219, Jan 2015.

III Antikainen M., Aura T., Särelä M. Spook in Your Network: Attacking an SDN with a Compromised OpenFlow Switch. In *19th Nordic Conference (NordSec 2014)*, LNCS vol. 8788, pp. 229–244, Oct. 2014.

IV Kandoi R., Antikainen M.. Denial-of-Service Attacks in OpenFlow SDN Networks. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1322–1326, 2015.

V Ranjbar A., Antikainen M., Aura T. Domain Isolation in a Multi-Tenant Software-Defined Network. In *IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 16–25, Dec. 2015.

VI Antikainen M., Aura T., Särelä M. Denial-of-Service Attacks in Bloom-Filter-Based Forwarding. *IEEE/ACM Transactions on Networking*, vol. 22, no. 5, pp. 1463–1476, Oct. 2014.

Author's Contribution

Publication I: "Commitment-based device pairing with synchronized drawing"

The two first authors of this paper were the main authors. The author of this thesis was mainly responsible for the protocol design and analysis, as well as for the background research relating to the pairing protocols.

Publication II: "Commitment-based device-pairing protocol with synchronized drawings and comparison metrics"

The two first authors of this paper were the main authors. The author of this thesis was mainly responsible for the protocol design and analysis, protocol related background research, and editing of the paper.

Publication III: "Spook in Your Network: Attacking an SDN with a Compromised OpenFlow Switch"

The author of this thesis was the principal author and wrote the paper.

Publication IV: "Denial-of-Service Attacks in OpenFlow SDN Networks"

The author of this thesis supervised the principal author. The contribution consisted of helping to define the problem, providing feedback on the work, and editing the paper.

Publication V: "Domain Isolation in a Multi-Tenant Software-Defined Network"

The author of this thesis supervised the principal author in writing the publication. The contribution consisted of providing feedback on the design and editing the paper.

Publication VI: "Denial-of-Service Attacks in Bloom-Filter-Based Forwarding"

The paper was a collaborative effort between the authors. The author of this thesis wrote the first draft of the paper, was the main contributor to sections 3–5, and participated in editing the final paper.

1. Introduction

Computer networking has changed greatly from what is used to be few decades ago. For a layman, one of the most visible changes has been the rapid increase in the number of Internet-connected devices as our surroundings get filled with various kinds of tiny computers. This trend is also likely to continue – the number of Internet-connected devices is expected to double every 2-3 years [6, 7]. A large fraction of this growth comes from *Internet-of-Things* (IoT) devices that are constrained in their computation and storage capabilities, communicate wirelessly, and often lack human input methods. Another recent trend in networking, less visible to layman, is the emergence of software-defined networking (SDN) [55], which centralizes the decision making in traditionally decentralized networks and thus eases the development and testing of novel network applications. Both of these developments, IoT and SDN, have received plenty of attention from researchers as well as from the industry because they are believed to be the key concepts that will shape the future networking [90, 13].

As most developments in computing, also IoT and SDN have their background in the continuous search of new applications and lower product development and operational costs. IoT connects the ubiquitous computers to each other and to the cloud and thus enables novel applications that were not possible before. On the other hand, one of the main drivers behind SDN was to reduce the cost and complexity of developing new network applications. However, IoT or SDN not only create opportunities, but also shape the security environment and introduce new threats to the systems. As an example, unlike traditional computers, IoT devices have limited CPU, memory, and battery capacity, which makes it more difficult to implement security protocols on them. Also, because of the small form factor, they often lack proper input devices which makes configur-

ing them more difficult. Likewise, SDN centralizes the network's control plane, which has traditionally been distributed, and thus radically alters the threat environment. These changes in the threats must be carefully studied early on in the product development cycle to ensure that possible design or implementation flaws do not spread widely.

Authentication is one of the main concepts when studying the security of the emerging technologies. Authentication is basically the act of verifying a communication peer's claims about its identity or other attributes [109]. Another concept closely related to authentication is *authorization*. While authentication is the act of verifying that a claimed attribute is true, authorization means verifying that an entity is permitted to take an action [79]. Traditionally, authorization comes after identity authentication [1], but in modern communication systems, these concepts are often intertwined. Firstly, in device pairing, devices can only authenticate each other after the user has authorized their association with each other. Secondly, if the potential threat comes from an insider such as another user of the network or even a compromised network element, logical isolation may take the place of both authentication and authorization. Thirdly, when fighting unauthorized traffic, authentication is typically a part of the solution, but does not alone prevent denial-of-service attacks. This thesis studies a set of such convoluted security problems in network and communication security, and proposes some solutions. These problems arose in several different research projects related to the security of the Internet of Things and future networking technologies.

1.1 Structure of the thesis

This dissertation consists of an introduction part and six original peer-reviewed publications. The common factor in the publications is network and communication security in IoT and SDN. The introduction is split into two parts as follows.

- Chapter 2 summarizes Publications **I**–**II**. It discusses authenticated key-exchange protocols for establishing secure wireless communication channel between two devices. More specifically, it describes a novel commitment-based key-exchange protocol for Internet-of-Things (IoT) devices. The protocol uses a shared fuzzy input for authenticating a negotiated session key. Unlike previous work, the protocol does not make

any assumptions on the entropy distribution of the fuzzy secret.

- Chapter 3 summarizes Publications **III**–**VI**. It moves the focus to the security of network infrastructure and, more specifically, software defined networks. In addition to giving an introduction to the broad topic of *SDN security*, we discuss the importance of device authentication in SDN in a situation where an attacker already has a foothold in the target network (Publication **III**). Another class of attacks that is discussed in detail is denial-of-service (DoS) attacks that originate from the data-plane (Publication **IV**). In addition to explaining security challenges that software-defined networks face, the chapter also describes a novel tenant isolation mechanism that can be implemented with SDN technologies (Publication **V**). The chapter ends by presenting a security analysis of several capability-based multicast-forwarding schemes that have been proposed for SDN environments (Publication **VI**). This analysis shows that the proposed protocols use a flawed path-authorization mechanism which makes them vulnerable to various DoS attacks.

1.2 Methodology

This thesis uses the well-established methodology in the field of systems and networking security research. This includes prototyping, which is used to verify new designs (Publications **I**, **II**, and **V**). In addition to prototyping, the developed protocols were verified with an automatic protocol verifier to avoid crude mistakes (Publications **I** and **II**). The attacks presented in this thesis are tested either in physical (Publication **IV**) or emulated environments (Publication **III**). Attacks presented in Publication **VI** were found with analytical means and their severity was evaluated with simulations and mathematical modeling.

2. Authenticated key-exchange with shared fuzzy secret

This chapter describes a protocol for authenticated key-exchange with fuzzy shared secret. The protocol was developed to enable pairing between Internet-of-Things (IoT) devices that have a touch-screen or other touch-sensitive surface. This work was originally published in Publication **I** and **II**. While the author of this thesis took part in the discussions of all parts of the papers, we will focus here on the security protocol design, which is the author's main contribution to the team effort.

The number of network connected devices has seen a huge increase in the past and this trend is likely to continue into the foreseeable future [6, 7]. Many of the new devices can be described as Internet-of-Things devices. These devices are constrained in terms of CPU, memory, and battery. They also have limited user-interfaces and typically communicate wirelessly with protocols such as Wi-Fi [5], Bluetooth [2], or ZigBee [3]. Establishing a secure wireless channel between two IoT devices that have no prior information about each other is a widely studied problem called *secure device pairing* [86].

Wireless channels are particularly interesting from the security point of view because the wireless communication is comparably easy to eavesdrop and even manipulate without being detected. Unfortunately, it is not trivial to securely set up a wireless channel between two devices if they initially know nothing about each other. Without the a-priori information, such as identifiers or keys, the devices must rely on out-of-band communication for identifying the right peer device and for creating secure channel with it. For example, the user may be required to carry some information, such as PIN-code, from one device to the other. This process becomes easily cumbersome for the user especially if the devices lack proper screens or keyboards.

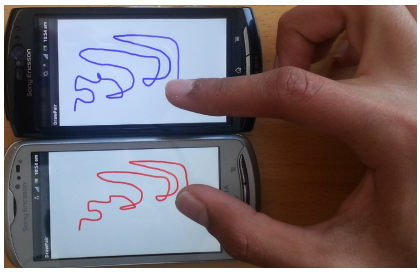

Figure 2.1. Inputting the fuzzy secret.

There have been many proposals that aim to reduce the amount of manual interaction required in the device pairing. These proposals include using contextual or location-dependent information, such as ambient sound or radio signals [62]. (See Chong et al. [30] for a comprehensive survey on different sources from which the secret can be extracted). Other mechanisms, on the other hand, require the user to help by producing the shared context information, for example, in the form of biometric input [23], simultaneous button presses [97], or simultaneous shaking of the devices [18, 63]. The common denominator for all of these proposals is that the shared secret information is *fuzzy* and can only be known approximately. As an example, although two recordings taken from the same ambient sound will have common features, the recordings will not be exactly the same. Another property for this kind of fuzzy information is that it is usually very difficult to estimate the entropy of such secret. These properties must be taken into account when designing a pairing mechanism and, specifically, when designing the security protocol for it.

Our study of the pairing protocols was motivated by the invention of a novel way for inputting the fuzzy secret: the user synchronously creates drawings on two touch-sensitive surfaces with two fingers of the same hand (see Fig. 2.1 for an illustration). The device pairing method works with two touch-screen or touch-pad devices including mobile phones, touch-pad mice, tablet computers, and more traditional IoT-like devices such as smart screens [100]. Similar method has later been proposed for user authentication [92].

After studying the pairing protocols for our novel pairing mechanism, we came to the conclusion that none of the existing protocols was suitable for this particular use case. More specifically, several of the earlier protocols either had too large communication overhead or made strong assumptions about the entropy of the secret (either about the total entropy or how the entropy is distributed in the secret data). This led us to de-

velop of a novel pairing protocol. Unlike the previous protocols, the new protocol makes only minimal assumptions about the entropy of the secret. This is important as measuring the entropy of fuzzy data is notoriously difficult and even a small amount of leaked data can reveal a lot about the fuzzy secret [46].

The rest of this chapter is split into three parts. We start by defining the threat model and reviewing the related work in Section 2.1. Then, Section 2.2 describes the design principles for our protocol as well as how the protocol was evaluated. After this, Section 2.3 discusses the protocol and concludes the chapter.

2.1 Background

This section reviews the previously proposed protocols for authenticated key establishment with fuzzy data. More specifically, the focus is on the situation where two devices want to negotiate a session key for a primary wireless communication channel (e.g. Bluetooth) while having no a-priori knowledge about each other. In addition to the primary channel, both devices have access to a low-throughput out-of-band (OOB) channel, which has a high error rate. The errors on this channel may be caused by variations in the sensory inputs or external noise, which causes the input to be fuzzy.

The goal of the protocol is to negotiate a session key that is used to secure the primary channel. The attacker has Dolev-Yao capabilities [36] on this channel and tries to access the communication either by eavesdropping or impersonating one or both devices (i.e. man-in-the-middle attack or one-sided impersonation attack). Unlike the primary channel, the OOB channel is assumed to have some inbuilt security that protects the integrity and confidentiality of the data. This may be, for example, because the OOB-channel is under direct supervision of the user or is highly location limited.

The protocols for solving this problem can be divided into three categories. Firstly, there are commitment-based protocols where an unauthenticated session key, which derived for example with Diffie-Hellman protocol, is authenticated with the commitments. Another possibility is to extract a secret key from the shared environment. This key can then be used either directly as a session key or to authenticate one. The third category is formed by the protocols where a key or authentication code is

transmitted over the OOB channel. We now briefly describe some protocol examples from each of these categories.

Commitment-based protocols

Commitments have been used for long to protect the communication integrity – probably the earliest protocol was Interlock proposed by Rivest and Shamir in 1984 [77]. Although this proposal used encryption functions rather than cryptographic hashes, the principles are the same as with modern protocols (see Hoang et al. [69] for a survey of modern commitment protocols).

There are three phases in commitment-based authenticated key exchange. First, in the *key-exchange phase*, the devices perform an unauthenticated key-exchange with Diffie-Hellman or by using public-key encryption without certificates. This produces a shared but still unauthenticated session key k. In the *commitment phase*, the devices cryptographically tie this key to a (short) shared secret p that they have received over the OOB-channel. Typically the commitment is a cryptographic hash $H(k, p, r)$, where r is a fresh random number that blinds the secrets. Finally, in the *opening phase*, the devices reveal the values r and p.[1] If the revealed shared secrets match and if the devices can recalculate each other's commitments using the just revealed information, the session key is deemed authentic.

The timing of the protocol phases is crucial to the security of the protocol. More specifically, neither device may move to the opening phase before the other has finished the commitment phase. Otherwise a man-in-the-middle (MitM) attacker can learn the shared secret p from one device and use it to compute the commitment for the other.

A clever way for enforcing the strict separation of the commitment and opening phases is shown by MANA III and its variants [4, 56, 101]. Here, the shared secret p is split into n pieces $p_1|p_2|\ldots|p_n$. The protocol then performs a separate commitment and opening for each round so that each round starts only after the previous ends. While this allows the attacker to learn one part p_i (typically p_1) of the shared secret, it cannot learn more unless it manages to guess a commitment correctly. The attacker can, nevertheless, use this learned information when performing the key-exchange with the other device (i.e. one-sided impersonation). Thus, given that the shared secret k has e_k bits of entropy, the protocol utilizes $e_k \cdot \frac{n-1}{n}$

[1] If the shared secret is not fuzzy (e.g. a PIN-code) there is no need to reveal p.

bits out of e_k. Increasing the number of protocol rounds obviously helps, but it also increases the communication overhead as the required number of round trips grows.

As mentioned above, commitment protocols work with fuzzy as well as with non-fuzzy secrets. The only difference with fuzzy secret is that the secret must be revealed alongside the random number r. Because of this, MANA III-like commitment protocols can be found from several protocols that use fuzzy information as the shared secret. As an example, Amigo [105] uses the ambient radio environment as the shared secret while Soriente et al. [97] have the user press buttons on both devices simultaneously and use the timing of the button presses as the shared secret.

However, the challenge with MANA III-like protocols is how the fuzzy secret p is split into parts. Firstly, the parts on both devices must correspond to each other. Secondly, the security of these protocols is based on the assumption that every part is independent from the others. In other words, secret p_i must not convey any information about piece p_j, $j > i$. Thirdly, all the parts must contain approximately the same amount of entropy. These requirements are difficult to meet when dealing with fuzzy secrets.

There are also other ways to enforce the separation of the commitment and opening phases. Roscoe [81] recently presented an approach where the phases could be separated with a time-lock – i.e. a computational puzzle that cannot be solved within certain time [78]. Our protocol follows the commitment approach but instead relies on implicit time synchronization from a user action.

Extracting key from noisy environment

Another solution is to extract a secret key directly from the environment shared by the devices. This naturally produces a fuzzy secret because environmental measurements are inherently noisy. If the key obtained this way contains enough entropy, it could be used directly as the session key. However, as it is usually difficult to estimate the total entropy in the fuzzy data, it usually is a better to use the fuzzy secret to authenticate a key exchange such as Diffie-Hellman.

While the general idea behind this approach is simple, it can be challenging to extract exact keys from fuzzy measurements. Most commonly, this is achieved with *fuzzy cryptography*. These protocols were originally

developed for biometric authentication and have later been used in device pairing [23, 24, 68]. While the protocol details vary, the general principle is that the paired devices exchange error correcting codes for their measurements – in clear over the insecure primary channel [15, 35]. Using this error correcting information, the devices may calculate an exact key from their own fuzzy measurements. The data-sources that have been proposed so far include biometrics [24, 23], ambient audio [96, 87, 68], and environment luminosity [68].

The challenge with fuzzy cryptography is to design the error correction scheme in such a way that the error correction information that is sent over the insecure channel does not leak any information about the actual data. This can be difficult to achieve as it requires careful studying of the information content of the fuzzy data [46]. This is something we want to avoid in our protocol.

In addition to the fuzzy-cryptography based approaches, some proposals try to create a secret key by extracting key features from the fuzzy data. The potential differences in the measured features are then tackled either by using only the most significant features from the data (e.g. ShakeMe [111]), or simply by not considering unequal features (CKP [64], SAPHE [44], and NFPA [58]). The inherent problem with these seemingly simple key-extraction methods is that the effective key strength of the key will be small even if a lot of entropy would be available – Yüzügüzel et al. [111] reported that they could extract *on average* only 15 bits of entropy from five seconds of device shaking. This is because only features that are guaranteed to match between the two devices can be used when creating the key.

Key distribution over noisy channel

Finally, some protocols rely on sending a secret key or an authentication code over a noisy OOB channel. These protocols offer a simple and secure way for authenticated key exchange assuming that the OOB channel cannot be eavesdropped and that active attacks can be detected. Human Assisted Pure Audio Pairing (HAPADEP) illustrates this approach [98]. There, an encryption key is sent over the audio OOB channel together with an error correction code. Alternatively, it is possible to use the OOB channel for verifying the result of a key exchange, as has been shown by BEDA [97], Loud and clear [42], and Seeing-is-believing [65]. Sending an authentication code over the OOB-channel (rather than the whole key) is

usually preferable as this makes the key exchange resistant to passive attacks on the OOB channel. Some protocols, such as EAP-NOOB [14], combine both of these methods and send the encryption key as well as authentication code over the OOB channel.

2.2 Time-based commitments and authenticated key-exchange with fuzzy secret

This section describes the design principles behind our pairing protocol. The details of the protocol can be found in Publications **I** and **II**.

The protocol was developed for a pairing mechanism that extracts the fuzzy secret from drawings that the user simultaneously creates on two devices using two fingers from the same hand. Typically the user does this by holding the devices on one hand and making the drawings with the thumb and index finger of the other hand. This makes the inputted drawings fuzzy – valid drawings are similar, but not exactly equal.

The main reason for developing a new pairing protocol instead of using for example some of the protocols reviewed in the previous section was that we had no reliable way of estimating how much entropy there is in the drawings that the user creates. (More recent research has provided evidence that the entropy of free-form drawings slightly exceeds the entropy of typical PIN-codes [92]). More importantly, we did not know how the entropy is distributed in the drawing, or in other words, would it be possible for the attacker to predict the whole drawing based on some part of it. Because of this uncertainty, the pairing protocol should not leak any information about the content of the fuzzy secret to a potential attacker.[2] This rules out protocols based on fuzzy cryptography as well as MANA III-like protocols with few protocol rounds. While MANA III -like protocol with several rounds would address the problem related to the entropy, this would increase the communication overhead. Directly extracting keys from the fuzzy input (without the use of fuzzy cryptography), on the other hand, would reduce the effective entropy of the shared secret, which renders this approach unacceptable. These practical problems motivated us to develop a new protocol where the secrets are communicated and revealed in one piece.

Our protocol uses commitments to ensure the authenticity of the session

[2]Strictly speaking, our protocol does leak the length of the secret, but nothing about its content.

key. Recall from the previous section that a commitment-based protocol has three phases: (1) unauthenticated key exchange, (2) commitment phase, and (3) opening phase. Moreover, recall that the security of such protocol relies on the strict separation of the commitment and opening phases. We developed two protocol variants that follow this same structure. In the first *explicit variant*, the user signals the devices when both of them have received commitments. In practice, this requires that the devices indicate that they have received the peer's commitment and that the user allows them to continue to the opening phase after both devices have reached this state. After this, the devices may reveal the secrets and finish the protocol.

The drawback of the explicit protocol variant is that it requires the user not to press the "continue"-button on one device before the other device has received its commitment. If this did not happen, a man-in-the-middle attacker would be able to launch an impersonation attack against one of the two devices. While it may be possible to educate the users to operate the pairing mechanism securely, the risk of users mistakes cannot be completely removed.

In the *implicit protocol variant*, the user is not required to signal the movement from the commitment phase to the opening phase. Instead, the roughly simultaneous phase transition is achieved with synchronized clocks. More specifically, the devices store the time when the user stops drawing and lifts fingers from the touch sensitive surface. This *reference event* happens roughly simultaneously on both devices and can be used to synchronize the devices' internal clocks. With information about this reference event, the devices can then progress in the protocol roughly synchronously. The minor synchronization error are tackled by allowing small deviation in the timings of the reference event. In our prototype, this margin of error was set to 500 ms. Too short drawings, on the other hand, were prevented in the prototype by requiring the drawing to be at least 4 seconds long. This value, however, could be much smaller – in a recent work, Sherman et al. [92] show that the length of a free-form drawing correlates only very weakly with its entropy and that roughly 2 seconds of drawing produces entropy that is comparable to a PIN-code.

The security of both protocol variants was verified with ProVerif [19] automated cryptographic protocol verifier (the used ProVerif-model can be found in the appendix of Publication **I**).

2.3 Discussion

The developed commitment protocol has three main limitations. Firstly, the protocol requires that the devices can be synchronized based on some reference event. In other words, the protocol cannot be used with inputs that do not provide such information that can be used for synchronizing the devices. Secondly, because the protocol uses commitment, the fuzzy-secrets must be revealed during the pairing process. Thus, this protocol should not be used with long-term secrets such as biometric data. (Our protocol does encrypt all commitment-related messages with the unauthenticated session key. While this prevents the information from leaking to a passive eavesdropper, it does not protect long-term secrets against active impersonation attacks.)

Finally, the synchronization of the devices relies on user actions. Failing to synchronize the devices opens possibilities for a man-in-the-middle attacker. We did pay a special attention to this and augmented the prototype with mechanisms that, for example, detect if the user accidentally lifts one finger off the device. However, the risk of erratic user behavior cannot be completely removed.

On the positive side, the proposed timing-based commitment protocol has several benefits compared to other pairing protocols. Firstly, as already mentioned, it makes very few assumptions about the entropy of the fuzzy secret. Because the fuzzy secret is used only to authenticate a sessions key (instead of deriving a session key directly from the fuzzy secret), it does not need to contain much entropy, but if it does, the security against opening attacks is increased accordingly. Also, because the secret is not split in any way, no information about it is leaked to a potential attacker. Thirdly, the fuzzy secrets are be revealed to the other device as a whole. This enables the devices to use any distance to compare the drawings and allows independent development of the protocol and the distance metrics used to compare the drawings. Different distance metrics are discussed more detailed in Publication **II**.

3. Towards secure software-defined networks

Traditionally, computer networks have run in a distributed manner so that routers and other network devices implement well specified protocols that define the network's behavior. Software-defined networking (SDN) changes this by replacing the distributed decision making with a logically centralized *controller* and by unifying the interface through which the controller reaches the network switches. This makes the network application development, testing, and maintenance easier and cheaper [52]. While the key ideas behind SDN can be traced back to the active-networking research done in the 90s [103, 25], SDN did not gain significant momentum until OpenFlow was introduced in 2008[1] [66]. The change, largely initiated by the introduction of OpenFlow, is said to be singular in the history of networking [91]. This claim is further attested by the sheer amount of research papers the topic has engendered [53, 70, 38].

The basic concept behind SDN is easily explained. In traditional networks, routers have a dual role: Firstly, they have a role on the control-plane as they perform routing, quality-of-service, and other control-plane functions. Secondly, they are data-plane elements meaning that they participate in packet forwarding. The idea behind SDN is to decouple these two functions. This is achieved by removing control-plane functions from the routers and instead placing the control-plane logic to the logically centralized controller. As a result, the switches[2] will be only responsible for data-plane packet forwarding. The controller then interfaces with the switches through a standard *south-bound interface*, such OpenFlow [66], NETCONF [37], LISP [80], or P4 [20]. Additionally, the controller in-

[1]The term *software-defined networking* was coined later, in 2009. [67]

[2]As the control-plane functionality is removed from the routers, they become switches according to SDN terminology. This notion is slightly inconsistent as even a traditional L2-switch may implement some control-plane functions, such as spanning-tree protocol (STP) and the SDN switches may forward packets based on L3 or L4 identifiers.

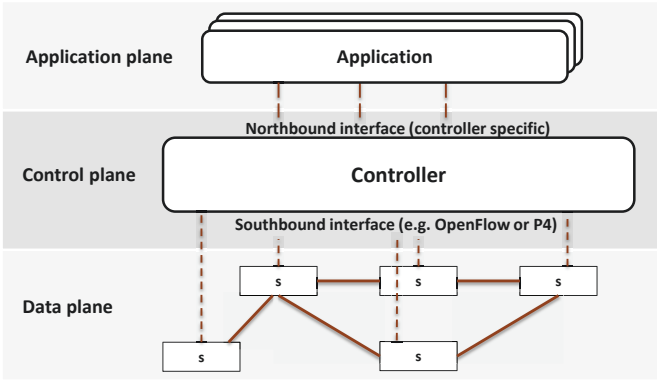

Figure 3.1. SDN architecture and terminology

terfaces with SDN applications through a controller-specific *north-bound interface*. This architecture shields the SDN applications from the problems caused by distributed state and makes it possible to create applications that feed requirements to the controller, which the controller then composes into packet processing rules for the data-plane [88]. Figure 3.1 illustrates the overall SDN architecture. Due to its popularity, this chapter mostly focuses on OpenFlow software-defined networks. The analysis, however, is for the most part generalizable to networks using other southbound protocols.

It is not surprising that an architectural change on this scale also changes the network security and threat landscape. The most obvious change is that more attention must be paid to ensure the security and robustness of the centralized controller which becomes a single point of failure. Also, securing the southbound communication (OpenFlow connections) becomes a priority because the ability to spoof OpenFlow messages would create almost unlimited possibilities for an attacker. In addition, the standardization of the southbound protocols will increase device and software homogeneity [54]. Thus, it is likely that design and implementation flaws have a broader impact in SDN than in traditional networks, which have had inherent protection in the form of decentralization, proprietary software, and heterogeneity of the network devices [54]. These observations motivated our research presented in Publications **III** and **IV**, which study denial-of-service attacks against OpenFlow networks and attacks that originate from compromised network elements.

That said, while SDN radically changes the threat landscape, it also creates several new possibilities. Novel network applications that have been

developed with SDN technologies range from SDN-based denial-of-service prevention mechanisms [110] to traffic engineering [8]. Applications that will be discussed more in this thesis include tenant isolation (Publication **V**) and source-routed multicast mechanisms (Publication **VI**).

This chapter summarizes the work originally done in Publications **III**–**VI**. First, Section 3.1 gives a brief survey to SDN security. Special attention is given to attacks that originate from compromised network elements and to denial-of-service (DoS) attacks originating from the data-plane, which are the topics of Publications **III** and **IV**, respectively. Section 3.2 takes more positive view on SDN and discusses Publication **V**, which describes a mechanism with which OpenFlow can be used to isolate domains in a multi-tenant network. Finally, Section 3.3 summarizes Publication **VI** which provides a security analysis of a source-routed multicast protocol that has been proposed for software-defined networks as well as earlier designs for the future Internet. The results show that the proposed protocol has inherent vulnerabilities that render it unsuitable for open networks.

3.1 Security challenges in SDN

The prior work on how to secure software-defined networks is plentiful [88, 59]. A commonly used way of approaching this topic to treat the security of the application and control planes, southbound interface (i.e. OpenFlow channel), and data plane separately [54, 59]. This section uses the same approach. The focus will be on attacks targeting the OpenFlow communication and on the data-plane security as these relate closely to Publication **III** and **IV**. Other aspects of SDN security are discussed only briefly (see Li et al. [59] for a recent and more comprehensive survey on the topic). In the rest of this chapter, we assume that the reader knows the basics of SDN and OpenFlow. Those unfamiliar with them are referred to Kreutz et al. [53].

Application plane and controller security

It is not surprising that a bulk of the SDN security research has focused on protecting the control plane. After all, SDN mainly changes how the network's control plane functions. Also, the brain of the network lies on the control plane making it a tempting target for adversaries.

One perspective to the control-plane security is how to reduce the risk of an unauthorized controller access. Li et al. [57] demonstrate this approach with an architecture where a network with multiple controllers support Byzantine fault tolerance. There are also other distributed controller designs that emphasize fault-tolerance [22] and distribution [104, 47]. While security is not the main motivation behind these proposals, increased distribution usually also yields resilience and can reduce the risk of a single controller becoming compromised.

Another control-plane security problem relates to the applications – how the controller can be protected against ill-behaving applications and how the rules that the controller receives via the northbound interface can be validated. Inconsistencies and interoperability issues are possible because the network applications above the north-bound interface may be provided by different vendors and may not be trusted as much as the actual controller. As an example, an attacker who can alter the behavior of some application may try to create several rules where no single rule alone violates the policies, but together they create unwanted behavior. This problem has led to the development of security enforcement kernels which enhances the controller with security functions such as access-control, flow-rule conflict resolution, and security auditing services [73, 74]. Other approaches that try to address the problem of incorrect policies or flow-rules utilize model checking [26], symbolic execution [51], or other real-time verification mechanisms [10].

OpenFlow security

The security of the southbound interface, namely the OpenFlow, has also gotten plenty of attention from the researchers. It is easy to see, that an attacker able to become a man in the middle in the OpenFlow connection would effectively get control over the OpenFlow switch. Just the capability to eavesdrop OpenFlow communication may provide valuable information to the attacker [94]. These, relatively obvious threats, are addressed in the OpenFlow standard by allowing the OpenFlow connection to be run over TLS (DTLS is possible for auxiliary OpenFlow connections) [72]. However, rather surprisingly, the use of TLS is optional according to the standard, and the TLS adoption has been notably slow [84].

Slow adoption of TLS is not the only problem that relates to securing the OpenFlow channel. In the context of SDN, the proper use of TLS requires that both the switch and the controller are mutually authenticated which

can only be achieved if the administrators distribute key pairs and certificates for every network device [89]. This can be considered as a nuisance by the administrators who may be tempted to take a shortcut and only use TLS with only one-sided authentication or, in the worst case, simply opt-out from the use of TLS [16]. Any shortcuts with TLS, however, may lead to severe security problems as pointed out by the IETF [108]. We made the same observation in Publication **III** where we analyze what kinds of attacks are possible if an attacker has already gotten a foothold in the target network and has compromised one of the switches. We found that the consequences of such an attack highly depend on whether the network uses TLS to protect the controller-plane traffic, whether the network uses TLS with mutual authentication, and whether the OpenFlow traffic is physically isolated from the data-plane. In the worst case, the attacker can spoof the state of several switches and even to perform topology spoofing. These spoofing attacks have later gotten more attention from the research community [45, 34] as well as the threat model where all network elements, and SDN-switches in particular, are not considered trusted [29, 41].

Data-plane security

While the majority of the changes that OpenFlow brings to traditional networking relate to the control-plane, there may also be vulnerabilities that are exploitable through the data-plane. Before going into the details, it is instructive to take a look at the different ways in which OpenFlow networks can be configured.

There are two different ways of routing the OpenFlow connections. Firstly, the network may be structured so that there is a dedicated physical control network that is used for control-plane signaling, namely for OpenFlow. This is called an *out-of-band* control-network. Alternatively, the control-plane may be logically isolated from the data-plane for example with VLAN tags. OpenFlow connections and data-plane traffic would, however, still travel over the same underlying physical network. This option is called an *in-band* control-channel. While the out-of-band channel is more palatable from the security point of view, creating a redundant network for control-traffic may not be always possible because of economical or practical reasons.

On a high level, there are two ways in which an OpenFlow network can be configured. Firstly, all of the forwarding rules, or *flow-table entries* in

the OpenFlow terminology, can be proactively configured on the switches. Alternatively, the flow-table entries can be installed after a switch sees a packet belonging to a new flow. In this latter, reactive model, when a switch sees a packet and does not know how to forward, it buffers the packet and sends the packet header to the controller. The controller then makes the forwarding decision on behalf of the switch, after which it installs a new flow-table entry to the switch in order to ensure the smooth forwarding of the following packets in the same flow. Unfortunately, the reactive flow-table rule installation may create an avenue for denial-of-service attacks. The attacker can flood the network with packets for which the switches do not yet have matching flow entries. This will cause the switches to constantly ask the controller for new flow-table rules, which eventually causes the flow tables and the switch buffers to become full. Suitable packet formats for the attack can be identified by measuring the packet round-trip times [94]. The use of an in-band control channel exacerbates this attack as the control-plane and data-plane share the bandwidth. We have studied these kinds of DoS attacks in Publication **IV**.

There are several ways to mitigate the threat of such DoS attacks. Firstly, the network should be configured so that it uses generic match rules in the flow-table entries. As a general principle, the match rules should be enough long living and generic so that a potential attacker cannot cause new flow-table entries to be created simply by modifying the packet fields. This requires flow aggregation and finding the optimal time-out periods for the flow-table entries. Rate limiting of OpenFlow messages may also help in mitigating the threat (this feature was introduced in OpenFlow 1.3.0 [71]). Finally, there have been proposals that involve first detecting a DoS attack and then trying to limit its impact, which may further mitigate the threat of these attacks [106, 95].

3.2 Tenant isolation with OpenFlow

So far this chapter has focused on security challenges. SDN, however, also creates possibilities for novel network applications. We now briefly describe an architecture for an OpenFlow-based multi-tenant environment. This work was originally described in Publication **V**.

In multi-tenant environments, a *service provider* wants to provide shared network resources for its *tenants*. This resource sharing brings several well known benefits such as better resource utilization as well as reduced

operational costs [17]. In a palatable architecture, the tenants should be able to control their own (leased) network resources while at the same time be isolated. This means that the tenants should be unable to affect other tenants. The tenants should, however, be able to open some of their services, so called inter-tenant services, to other tenants. The architecture presented in Publication **V** achieves this: while the tenants are isolated from each other by default, they may loosen up the isolation and open some of their services to other tenants.

Before going to the details of the architecture, let us take a look at a typical multi-tenant network in Figure 3.2. Firstly, every tenant in the network has dedicated (virtual) hosts, which are connected to a (virtual) edge switch. This means that a host can always be identified by the interface of the edge switch. While the hosts and their connections to the edge switch are dedicated to a single tenant, the core-network connecting the edge switches is shared.

Providing tenant isolation in a multi-tenant environment is essentially a problem about multiplexing and how the tenants' packets are identified in the shared infrastructure [31]. In general, there are three ways in which this can be done. Firstly, the network can be sliced meaning that, for example, the available address space is split between the tenants (see e.g. [93, 32]). While this is a very simple way of achieving tenant-isolation, it decreases the IPv4 address space size available to the tenants. Second possibility is to use tunneling in the shared network. In this case, the edge switches simply encapsulates all packets going towards the shared core network and decapsulates packets arriving from there. The encapsulation must include sufficient information for forwarding the packet through the shared network as well as a tenant identifier that allows the egress switch to forward the packet to the correct destination host. Such tunneling solutions can be found from several multi-tenant related architectures [33, 60, 107] as well as from other, not strictly multi-tenant related, solutions [39, 61, 43].

The third way to multiplex packets through the shared core network

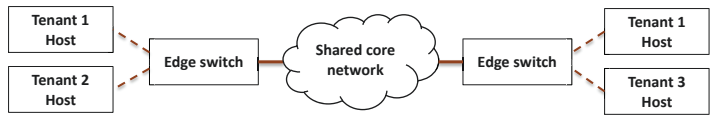

Figure 3.2. A typical multi-tenant network architecture. Every virtual host are connects to a virtual edge switches, which are connected via the shared core network.

is to use address rewriting (see e.g. [9]). As mentioned above, when the packet is sent to the core network, it must be augmented with the information needed for forwarding the packet through the core, and with information that uniquely identifies the destination. This information can also be rewritten to the original header without adding a new header. In the architecture presented in Publication **V**, we use the Ethernet source and destination addresses for this. The main benefit of address rewriting compared to encapsulation is that it does not increase the packet size, which could in some scenarios lead to fragmenting [60], and, as discussed in the publication, is typically slower compared to packet rewriting. The drawback of rewriting is that, unlike encapsulation, it requires that the packet's L2 header to stay unmodified in the shared infrastructure.

The inter-tenant communication in our proposed architecture is solved by dedicating a some private IP addresses for inter-domain services. While this does limit the number of addresses available to the tenants, it is a more scalable solution than the use of public IP addresses. However, on a more general level, the way the inter-tenant communication is designed in the architecture is not tied to the forwarding mechanism but would also work if tunneling was used instead of packet rewriting.

3.3 Source-routed capabilities and their security

One of the promises of SDN was that it allows easy development of network applications and even protocols by developing these features as controller applications.[3] This has inspired research on alternative forwarding mechanisms. One such proposal is Bloom-filter based source-routed multicast [48]. While the research on Bloom-filter based forwarding precedes the SDN research [75], the emergence of SDN has sparked the academics' interest on the topic again. This is because SDN technologies provide natural and easy deployment paths for these protocols. At the time of writing, there are Bloom-filter forwarding implementations for networks using OpenFlow [27, 76], P4 [12], and NetFPGA [50, 40].

The idea behind Bloom-filter based forwarding is simple. First, consider the network as a directed graph $G = (V, E)$. Any multicast tree T in the

[3]While OpenFlow is relatively restricted, there are several other SDN technologies that enable packet matching and forwarding on the switches based on arbitrary fields [21, 20, 49]. These technologies give the developers more leeway when designing new protocols.

network can be depicted as a set of links $T \subseteq E$. (The tree reduces to a single path in unicast). The basic idea is that the tree T is included in the header of every packet sent to that tree. Based on the information found from the packet, the network switches can forward the packets towards the intended destination. The header size can be kept reasonable by encoding the set T as a Bloom filter.

This approach has several intriguing properties. First, Bloom filters allow the multicast trees to be represented compactly – typically the proposals use 256-bit Bloom filters which can store trees with around 32 links. Secondly, since the multicast tree is embedded in every packet, the switches do not have to store any per-group state. This is in stark contrast to IP multicast that requires routers to store state for every multicast group traversing through them.

The forwarding scheme, however, also has its problems. Bloom filter is a probabilistic data-structure and it is possible that a packet is forwarded to a link to which it was not originally intended. While a small number of false positives can be allowed (there are several tricks that can be used to reduce the probability for false positives [102]), there is always a risk that a packet is forwarded to a node that was earlier in the multicast tree. Without countermeasures, this would cause a potentially infinite loop, which an adversary could exploit for denial-of-service (DoS) attacks.

Rothenberg et al. [82] proposed solving both accidental and maliciously created forwarding loops by applying a pseudo-random bit-permutation on the Bloom filter at every hop. This would essentially make the Bloom filter a function of the traversed path and thus break potential loops.[4] The invention of this relatively brilliant looping countermeasure led its inventors to believe that the same mechanism would essentially make the Bloom filter a source-routed *capability*. Network capability is basically an approach to solve denial-of-service (DoS) attacks by allowing routers to check from a packet itself whether it has been authorized by a receiver [11]. Because the Bloom filter that the sender uses on a packet is composed of link identifiers that the sender is not supposed to know, and because the sender does not (presumably) know the permutations that the switches apply, it would seem that the Bloom filter could be used to as a capability. (Naturally, this requires that the network has some mechanism for end-hosts to obtain valid Bloom filters. Typically this is achieved

[4]While this countermeasure is probabilistic, infinite loops have been shown to be extremely unlikely when it is deployed [99].

with a logically centralized entity similar to the SDN-controller [48], or with some distributed mechanism such as the one presented by Särela et al. [85].)

In Publication **VI**, we show that the capability assumption is false – permuted Bloom-filters do not work as secure capabilities. Our analysis covers a wide range of different protocol variants and proves that an attacker can construct valid Bloom filters that can be used to send unsolicited traffic from a large number of compromised end hosts to a target node. In addition, the attacker can effectively prevent anyone from leaving a multicast group which they have once joined. While preventing these attacks is not impossible, most of the benefits of Bloom-filter forwarding would be lost if the effective but expensive countermeasures for these attacks were introduced. However, Bloom-filter based forwarding is still an option for trusted environments. It could, for example, be used inside data-centers (see e.g. [83, 27, 28]) or as a layer 2.5 protocol instead of MPLS (see e.g. [112]). Nevertheless, the results presented in Publication **VI** can be seen as a reminder that novel SDN-based applications do not always bring the benefit they promise.

4. Conclusion

Internet of things (IoT) and software-defined networking (SDN) are expected to have a great impact on future networking. As a consequence, they will also alter the networks' security environment and introduce new threats. This dissertation is composed of six original publications that discuss various aspects of the security of IoT and SDN. A special emphasis is put on authentication and authorization – the IoT-related part of the dissertation discusses device authentication and pairing while the SDN-related part discusses topics such as authentication in OpenFlow, isolation in multi-tenant SDN networks, and SDN-based source-routed capabilities.

The main results and outcomes presented in this dissertation are the following. Firstly, Publications **I**–**II** describe a novel protocol for device pairing and authenticated key exchange. Unlike previous protocols, it allows devices to be securely paired with a shared fuzzy secret while not making any assumptions on how entropy is distributed in the secret. After this, we study various aspects of SDN security in Publications **III**–**VI**. Focus is on the security challenges in OpenFlow and, more specifically on the device authentication (Publication **III**) and denial-of-service attacks (Publication **IV**). These publications emphasizes the importance of device authentication in SDN networks and the DoS risk caused by badly configured OpenFlow devices. After discussing the security challenges of SDN, we take a more positive look at SDN security and describe an architecture for OpenFlow-based tenant isolation (Publication **V**). Finally, the dissertation is concluded with a security analysis of proposed capability-based multicast schemes for SDN environments (Publication **VI**). This analysis shows that the proposed protocols use a flawed path-authorization mechanism which makes them vulnerable to various denial-of-service (DoS) attacks and limits their use to closed networks.

The security of emerging networking technologies is a broad topic and there are still many open and undiscovered problems in the area. While this dissertation is a step on the path towards more secure and resilient networks, there is still a great demand for further research on this topic.

References

[1] *Trusted computer system evaluation criteria CSC-STD-00l-83*. Department of defence standard. Department of defence, Aug 1983.

[2] IEEE std 802.15.1-2005 "Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)", 2005. IEEE Computer Society.

[3] ZigBee specification. pages 344–346, 2006. Zigbee Alliance Document 053474r13.

[4] ISO/IEC 9798-1:2010 Information technology – Security techniques – Entity authentication – Part 1: General. Technical report, ISO/IEC, 2010.

[5] IEEE std 802.11-2012, "Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications", 2012. IEEE Computer Society.

[6] Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015. Gartner, Press release, November 2015.

[7] 'Internet of things' connected devices to almost triple to over 38 billion units by 2020. Juniper Research, Press release, November 2015.

[8] Ian F. Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. A roadmap for traffic engineering in SDN-OpenFlow networks. *Computer Networks*, 71:1–30, 2014.

[9] Ali Al-Shabibi, Marc De Leenheer, Matteo Gerola, Ayaka Koshibe, William Snow, and Guru Parulkar. OpenVirteX: A network hypervisor. In *Open Networking Summit 2014 (ONS 2014)*, Santa Clara, CA, March 2014. USENIX Association.

[10] Ehab Al-Shaer and Saeed Al-Haj. Flowchecker: Configuration analysis and verification of federated OpenFlow infrastructures. In *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*, pages 37–44. ACM, 2010.

[11] Tom Anderson, Timothy Roscoe, and David Wetherall. Preventing Internet denial-of-service with capabilities. *ACM SIGCOMM Computer Communication Review*, 34(1):39–44, 2004.

[12] Markku Antikainen, Liang Wang, Dirk Trossen, and Arjuna Sathiaseelan. XBF: scaling up Bloom-filter-based source routing. *CoRR*, abs/1602.05853, 2016.

[13] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010. Elsevier.

[14] Tuomas Aura and Mohit Sethi. Nimble out-of-band authentication for EAP (EAP-NOOB). Internet-Draft draft-aura-eap-noob-01, IETF, July 2016. Work in Progress.

[15] Lucas Ballard, Seny Kamara, and Michael K Reiter. The practical subtleties of biometric key generation. *USENIX Security Symposium*, pages 61–74, 2008.

[16] Kevin Benton, L. Jean Camp, and Chris Small. OpenFlow vulnerability assessment. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 151–152, New York, NY, USA, 2013. ACM.

[17] C. P. Bezemer, A. Zaidman, B. Platzbeecker, T. Hurkmans, and A. '. Hart. Enabling multi-tenancy: An industrial experience report. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–8, September 2010.

[18] Daniel Bichler, Guido Stromberg, Mario Huemer, and Manuel Löw. Key generation based on acceleration data of shaking processes. In *Proceedings of the 9th International Conference on Ubiquitous Computing*, UbiComp '07, pages 304–317, Berlin, Heidelberg, 2007. Springer-Verlag.

[19] Bruno Blanchet, Ben Smyth, and Vincent Cheval. Proverif 1.86 pl3: Automatic cryptographic protocol verifier, user manual and tutorial, 2012.

[20] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, July 2014.

[21] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. *ACM SIGCOMM Computer Communication Review*, 43(4):99–110, August 2013.

[22] F. Botelho, A. Bessani, F. M. V. Ramos, and P. Ferreira. On the design of practical fault-tolerant SDN controllers. In *2014 Third European Workshop on Software Defined Networks*, pages 73–78, September 2014.

[23] Ileana Buhan, Jeroen Doumen, Pieter Hartel, and Raymond Veldhuis. Feeling is believing: a secure template exchange protocol. In *International Conference on Biometrics*, volume 4642 of *LNCS*, pages 897–906. Springer, 2007.

[24] Ileana Buhan, Jeroen Doumen, Pieter Hartel, and Raymond Veldhuis. Secure ad-hoc pairing with biometrics: SAfE. pages 450–456. Ubicomp 2007 Workshop Proceedings, 2007.

[25] Andrew T. Campbell, Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente, and Daniel Villela. A survey of programmable networks. *ACM SIGCOMM Computer Communication Review*, 29(2):7–23, April 1999.

[26] Marco Canini, Daniele Venzano, Peter Perešíni, Dejan Kostić, and Jennifer Rexford. A NICE way to test OpenFlow applications. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 127–140, 2012.

[27] A. B. M. Carlos, C. E. Rothenberg, and F. M. Maurício. In-packet Bloom filter based data center networking with distributed OpenFlow controllers. In *2010 IEEE Globecom Workshops*, pages 584–588, December 2010.

[28] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. V. Vasilakos. Survey on routing in data centers: insights and future directions. *IEEE Network*, 25(4):6–10, July 2011.

[29] Po-Wen Chi, Chien-Ting Kuo, Jing-Wei Guo, and Chin-Laung Lei. How to detect a compromised SDN switch. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–6, April 2015.

[30] Ming Ki Chong, Rene Mayrhofer, and Hans Gellersen. A survey of user interaction for spontaneous device association. *ACM Computing Surveys (CSUR)*, 47(1):8:1–8:40, May 2014.

[31] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Computer Networks*, 54(5):862–876, 2010. Elsevier.

[32] R. Doriguzzi Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori. Vertigo: Network virtualization and beyond. In *2012 European Workshop on Software Defined Networking*, pages 24–29, October 2012.

[33] B. Davie and J. Gross. A stateless transport tunneling protocol for network virtualization (STT). Internet-Draft draft-davie-stt-08, IETF, 2016.

[34] Mohan Dhawan, Rishabh Poddar, Kshiteej Mahajan, and Vijay Mann. SPHINX: Detecting security attacks in software-defined networks. In *Proceedings of the 2015 Network and Distributed System Security (NDSS) Symposium*, 2015.

[35] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, January 2008.

[36] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, Mar 1983.

[37] Rob Enns, Martin Bjorklund, Andy Bierman, and Jürgen Schönwälder. Network Configuration Protocol (NETCONF). RFC 6241, October 2015.

[38] Hamid Farhady, HyunYong Lee, and Akihiro Nakao. Software-defined networking: A survey. *Computer Networks*, 81:79–95, 2015. Elsevier.

[39] Dino Farinacci, Vince Fuller, David Meyer, and Darrel Lewis. The locator/id separation protocol (LISP). RFC 6830, January 2013.

[40] A Ghani and Pekka Nikander. Secure in-packet Bloom filter forwarding on the NetFPGA. In *European NetFPGA Developers Workshop*. University of Cambridge, Computer Laboratory, 2010.

[41] R. Ghannam and A. Chung. Handling malicious switches in software defined networks. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 1245–1248, April 2016.

[42] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: Human-verifiable authentication based on audio. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 10–10, 2006.

[43] Jesse Gross and Ilango Ganga. Geneve: Generic network virtualization encapsulation. Internet-Draft draft-ietf-nvo3-geneve-02, IETF, July 2016. Work in Progress.

[44] Bogdan Groza and Rene Mayrhofer. SAPHE: Simple accelerometer based wireless pairing with heuristic trees. *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, pages 3–5, 2012.

[45] Sungmin Hong, Lei Xu, Haopei Wang, and Guofei Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *Proceedings of the 2015 Network and Distributed System Security (NDSS) Symposium*, 2015.

[46] Tanya Ignatenko and Frans Willems. Information leakage in fuzzy commitment schemes. *IEEE Transactions on Information Forensics and Security*, 5(2):337–348, June 2010.

[47] Xin Jin, Li Erran Li, Laurent Vanbever, and Jennifer Rexford. SoftCell: Scalable and flexible cellular core network architecture. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 163–174, New York, NY, USA, 2013. ACM.

[48] Petri Jokela, András Zahemszky, Christian Esteve Rothenberg, Somaya Arianfar, and Pekka Nikander. LIPSIN: Line speed publish/subscribe inter-networking. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 195–206, New York, NY, USA, 2009. ACM.

[49] S. Jouet, R. Cziva, and D. P. Pezaros. Arbitrary packet matching in openflow. In *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6, July 2015.

[50] Jari Keinänen, Petri Jokela, and Kristian Slavov. Implementing zFilter based forwarding node on a NetFPGA. In *Proc. of NetFPGA Developers Workshop*, 2009.

[51] Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P Brighten Godfrey. Veriflow: Verifying network-wide invariants in real time. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 15–27, 2013.

[52] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, et al. Onix: A distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation, OSDI'10*, 2010.

[53] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, January 2015.

[54] Diego Kreutz, Fernando M.V. Ramos, and Paulo Verissimo. Towards secure and dependable software-defined networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, HotSDN '13*, pages 55–60, New York, New York, USA, August 2013.

[55] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM, 2010.

[56] Jan-Ove Larsson. Higher layer key exchange techniques for Bluetooth security. In *Open Group Conference, Amsterdam*, 2001.

[57] H. Li, P. Li, S. Guo, and A. Nayak. Byzantine-resilient secure software-defined networks with multiple controllers in cloud. *IEEE Transactions on Cloud Computing*, 2(4):436–447, October 2014.

[58] L. Li, X. Zhao, and G. Xue. A proximity authentication system for smartphones. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2015.

[59] Wenjuan Li, Weizhi Meng, and Lam For Kwok. A survey on OpenFlow-based software defined networks: Security challenges and countermeasures. *Journal of Network and Computer Applications*, 68:126–139, 2016.

[60] Mallik Mahalingam, T. Sridhar, Mike Bursell, Lawrence Kreeger, Chris Wright, Kenneth Duda, Puneet Agarwal, and Dinesh Dutt. Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks. RFC 7348, August 2014.

[61] F. Maino, V. Ermagan, Y. Hertoghs, D. Farinacci, and M. Smith. LISP control plane for network virtualization overlays. Internet-Draft draft-maino-nvo3-lisp-cp-03, IETF, October 2013.

[62] Suhas Mathur, Robert Miller, Alexander Varshavsky, Wade Trappe, and Narayan Mandayam. Proximate: proximity-based secure pairing using ambient wireless signals. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 211–224. ACM, 2011.

[63] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Transactions on Mobile Computing*, 8(6):792–806, June 2009.

[64] Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. volume 4480 of *LNCS*, pages 144–161. Springer Berlin Heidelberg, 2007.

[65] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *2005 IEEE Symposium on Security and Privacy (S P'05)*, pages 110–124, May 2005.

[66] N McKeown and Tom Anderson. OpenFlow: enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 69–74, April 2008.

[67] Nick McKeown. Software-defined networking. INFOCOM 2009 keynote talk, 2009. Slides available at: `http://www.cs.rutgers.edu/~badri/552dir/papers/intro/nick09.pdf`.

[68] Markus Miettinen, N. Asokan, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and Majid Sobhani. Context-based zero-interaction pairing and key evolution for advanced personal devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 880–891, New York, NY, USA, 2014. ACM.

[69] Long Hoang Nguyen and Andrew William Roscoe. Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey. *Journal of Computer Security*, 19(1):139–201, 2011.

[70] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634, Third quarter 2014.

[71] Open Networking Foundation. OpenFlow switch specification. Version 1.3.0. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf`, June 2012.

[72] Open Networking Foundation. OpenFlow switch specification. Version 1.5.0. `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf`, December 2014.

[73] Philip Porras, Seungwon Shin, Vinod Yegneswaran, Martin Fong, Mabry Tyson, and Guofei Gu. A security enforcement kernel for OpenFlow networks. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 121–126, New York, NY, USA, 2012. ACM.

[74] Phillip A Porras, Steven Cheung, Martin W Fong, Keith Skinner, and Vinod Yegneswaran. Securing the software defined network control layer. In *NDSS*, 2015.

[75] Sylvia Ratnasamy, Andrey Ermolinskiy, and Scott Shenker. Revisiting IP multicast. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 15–26, New York, NY, USA, 2006. ACM.

[76] Martin J. Reed, Mays F. Al-Naday, Nikolaos Thomos, Dirk Trossen, George Petropoulos, and Spiros Spirou. Stateless multicast switching in software defined networks. *CoRR*, abs/1511.06069, 2015.

[77] Ronald L Rivest and Adi Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–394, 1984.

[78] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, 1996.

[79] Dorothy Elizabeth Robling Denning. *Cryptography and Data Security*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1982.

[80] A. Rodriguez-Natal, M. Portoles-Comeras, V. Ermagan, D. Lewis, D. Farinacci, F. Maino, and A. Cabellos-Aparicio. LISP: A southbound SDN protocol? *IEEE Communications Magazine*, 53(7):201–207, July 2015.

[81] AW Roscoe. Detecting failed attacks on human-interactive security protocols. Technical report, Oxford University Department of Computer Science, 2016.

[82] Christian Esteve Rothenberg, Petri Jokela, Pekka Nikander, Mikko Sarela, and Jukka Ylitalo. Self-routing denial-of-service resistant capabilities using in-packet Bloom filters. In *Computer Network Defense (EC2ND), 2009 European Conference on*, pages 46–51. IEEE, 2009.

[83] Christian Esteve Rothenberg, C Macapuna, Fabio Verdi, Mauricio Magalhães, and András Zahemszky. Data center networking with in-packet Bloom filters. In *Proc. SBRC*, pages 553–566, 2010.

[84] Dominik Samociuk. Secure communication between OpenFlow switches and controllers. In *AFIN 2015 : The Seventh International Conference on Advances in Future Internet*, page 39, 2015.

[85] Mikko Särelä, Christian Esteve Rothenberg, András Zahemszky, Pekka Nikander, and Jörg Ott. *Information Security Technology for Applications: 15th Nordic Conference on Secure IT Systems, NordSec 2010*, chapter BloomCasting: Security in Bloom filter based multicast, pages 1–16. Springer Heidelberg, 2012.

[86] N. Saxena, J. E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *2006 IEEE Symposium on Security and Privacy (S P'06)*, pages 6 pp.–313, May 2006.

[87] Dominik Schurmann and Stephan Sigg. Secure communication based on ambient audio. *Mobile Computing, IEEE Transactions on*, 12(2):358–370, 2013.

[88] S. Scott-Hayward, G. O'Callaghan, and S. Sezer. SDN security: A survey. In *IEEE SDN for Future Networks and Services (SDN4FNS)*, pages 1–7, November 2013.

[89] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, 51(7):36–43, July 2013.

[90] S. Shenker. The future of networking, the past of protocols, October 2011. Presentation at Open Network Summit, Available at `http://www.youtube.com/watch?v=YHeyuD89n1Y`.

[91] Scott Shenker, M Casado, T Koponen, and N McKeown. The future of networking, and the past of protocols. Open Networking Summit (ONS) 2011 presentation. Online at http://www.opennetsummit.org/archives/apr12/site/talks/shenker-tue.pdf.

[92] Michael Sherman, Gradeigh Clark, Yulong Yang, Shridatt Sugrim, Arttu Modig, Janne Lindqvist, Antti Oulasvirta, and Teemu Roos. User-generated free-form gestures for authentication: Security and memorability. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 176–189, New York, NY, USA, 2014. ACM.

[93] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru M Parulkar. Can the production network be the testbed? In *OSDI*, volume 10, pages 1–6, 2010.

[94] Seungwon Shin and Guofei Gu. Attacking software-defined networks: A first feasibility study. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 165–166, New York, NY, USA, 2013. ACM.

[95] Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 413–424, New York, NY, USA, 2013. ACM.

[96] S Sigg, Y Ji, N Nguyen, and A Huynh. AdhocPairing: Spontaneous audio based secure device pairing for Android mobile devices. *Proceedings of the 4th International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use (IWSSI/SPMU'12)*, 2012.

[97] Claudio Soriente, Gene Tsudik, and Ersin Uzun. BEDA: Button-enabled device pairing. Cryptology ePrint Archive, Report 2007/246, 2007. Available at: http://eprint.iacr.org/2007/246.

[98] Claudio Soriente, Gene Tsudik, and Ersin Uzun. HAPADEP: Human-assisted pure audio device pairing. In *International Conference on Information Security*, volume 5222 of *LNCS*, pages 385–400. Springer, 2008.

[99] M. Särelä, C. Esteve Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott. Forwarding anomalies in Bloom filter-based multicast. In *INFOCOM, 2011 Proceedings IEEE*, pages 2399–2407, April 2011.

[100] G. S. Sundaram, B. Patibandala, H. Santhanam, S. Gaddam, V. K. Alla, G. R. Prakash, S. C. V. Chandracha, S. Boppana, and J. M. Conrad. Bluetooth communication using a touchscreen interface with the raspberry pi. In *Southeastcon, 2013 Proceedings of IEEE*, pages 1–4, April 2013.

[101] J. Suomalainen, J. Valkonen, and N. Asokan. Standards for security associations in personal networks: a comparative analysis. *International Journal of Security and Networks*, 4(1/2):87–100, 2009. Inderscience Publishers.

[102] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz. Theory and practice of Bloom filters for distributed systems. *IEEE Communications Surveys Tutorials*, 14(1):131–155, First quarter 2012.

[103] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[104] Amin Tootoonchian and Yashar Ganjali. HyperFlow: A distributed control plane for OpenFlow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pages 3–3, 2010.

[105] Alex Varshavsky and Adin Scannell. Amigo: Proximity-based authentication of mobile devices. *UbiComp 2007: Ubiquitous Computing*, pages 253–270.

[106] Haopei Wang, Lei Xu, and Guofei Gu. Of-guard: A DoS attack prevention extension in software-defined networks. *The Open Network Summit (ONS)*, 2014.

[107] Yu-Shun Wang and Pankaj Garg. NVGRE: Network virtualization using generic routing encapsulation. RFC 7637, September 2015.

[108] M. Wasserman and S. Hartman. Security analysis of the open networking foundation (ONF) OpenFlow switch specification. Internet-Draft draft-mrw-sdnsec-openflow-analysis-01, IETF, April 2013.

[109] Thomas YC Woo and Simon S Lam. *Authentication for distributed systems*. University of Texas at Austin, Department of Computer Sciences, 1991.

[110] Q. Yan, F. R. Yu, Q. Gong, and J. Li. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys Tutorials*, 18(1):602–622, First quarter 2016.

[111] H. Yüzugüzel, J. Niemi, S. Kiranyaz, M. Gabbouj, and T. Heinz. ShakeMe: Key generation from shared motion. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, pages 2130–2133, October 2015.

[112] A. Zahemszky, P. Jokela, M. Sarela, S. Ruponen, J. Kempf, and P. Nikander. Mpss: Multiprotocol stateless switching. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1–6, March 2010.

Errata

Publication I

Authors' organizational affiliation marked with dagger-symbol (†) should have been Department of Computer Science *and Engineering* instead of Department of Computer Science.

Page 5, last paragraph of section 3. The value of Δ_1 should be 200 ms and Δ_2 500 ms. Variable Δ_3 does not exist in the protocol or the prototype.

Citation [28] in the publication has incorrect publication year and venue. See the citation [87] in this dissertation for the correct information.

Publication III

The first column of Table 1 was not printed correctly. The table should look as shown below.

Table 1: List of attacks analyzed in this paper

	Flow-table modification				Control-channel hijacking			
	No TLS	Controller authentication	Mutual authentication	Out-of-band control channel	No TLS	Controller authentication	Mutual authentication	Out-of-band control channel
Eavesdrop control data with packet duplication	✓	✓	✓	✓	✓	✓	✓	✓
MitM attack by diverting traffic	✓	✓	✓	✓	✓	✓	✓	✓
Spoof the state of the compromised switch e.g. to avoid attack detection					✓	✓	✓	✓
Eavesdrop control traffic of the compromised switch					✓	✓	✓	✓
Eavesdrop downstream switches' control traffic	✓				✓			
Spoof downstream switches' state	✓				✓			
Topology spoofing: add bogus switches to the network	✓	✓		?[1]	✓	✓		?[1]
Empty downstream switches' flowtables (DoS)	✓				✓			

[1]Depends on the control-network implementation. This and many further attacks against uncompromised switches may be possible if the control network does not provide adequate isolation between switches.

Aalto-DD 275/2016

ISBN 978-952-60-7218-0 (printed)
ISBN 978-952-60-7217-3 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science
www.aalto.fi

BUSINESS +
ECONOMY

ART +
DESIGN +
ARCHITECTURE

SCIENCE +
TECHNOLOGY

CROSSOVER

DOCTORAL
DISSERTATIONS