Open Access

Open Journal of Internet of Things (OJIOT)
Volume 3, Issue 1, 2017
http://www.ronpub.com/ojiot
ISSN 2364-7108

Sensing as a Service: Secure Wireless Sensor Network Infrastructure Sharing for the Internet of Things

Cíntia B. Margi [A], Renan C. A. Alves [A], Johanna Sepulveda [B]

[A] Laboratory of Computer Networks and Architecture (LARC), University of Sao Paulo, Av. Prof. Luciano Gualberto, travessa 3, n.158, Sao Paulo, Brazil, {cintia, renanalves}@usp.br

[B] Chair of Security in Information Technology (EISEC), Technical University of Munich, Theresienstrasse 90, Munich, Germany, johanna.sepulveda@tum.de

Abstract

Internet of Things (IoT) and Wireless Sensor Networks (WSN) are composed of devices capable of sensing/actuation, communication and processing. They are valuable technology for the development of applications in several areas, such as environmental, industrial and urban monitoring and processes controlling. Given the challenges of different protocols and technologies used for communication, resource constrained devices nature, high connectivity and security requirements for the applications, the main challenges that need to be addressed include: secure communication between IoT devices, network resource management and the protected implementation of the security mechanisms. In this paper, we present a secure Software-Defined Networking (SDN) based framework that includes: communication protocols, node task programming middleware, communication and computation resource management features and security services. The communication layer for the constrained devices considers IT-SDN as its basis. Concerning security, we address the main services, the type of algorithms to achieve them, and why their secure implementation is needed. Lastly, we showcase how the Sensing as a Service paradigm could enable WSN usage in more environments.

Type of Paper and Keywords

Regular research paper: *sensing as a service, security, software-defined networking, Internet of Things*

1 Introduction

Research on Wireless Sensor Networks (WSN) started about 20 years ago, pushed by the development in the microelectronics industry as well as advances in wireless communications. Furthermore, the ability to collect data with increased spacial and temporal density increased the knowledge on events being monitored, which interested researchers and users of this collected data.

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2017)* in conjunction with the VLDB 2017 Conference in Munich, Germany. The proceedings of VLIoT@VLDB 2017 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

WSN has been used to support several different applications, mainly related to monitoring, detection and tracking. WSN nodes are typically battery-powered, resource constrained (i.e. limited amount of memory, processing capacity, bandwidth and range), and communicate through a multihop ad hoc network [6].

Routing protocols for WSN evolved over time. Initially most of the deployments had specific purposes and used routing algorithms tailored for their needs. For instance, directed diffusion was designed for event and interest based networks [15], while Collection Tree Protocol (CTP) is adequate for a traffic pattern from nodes to sink [11].

Considering the need to integrate Low power and Lossy Networks (LLN) to the Internet, Internet Engineering Task Force (IETF) formed working groups. IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) was specified (RFC 4944) and approaches to achieve header compression (RFC 6282) and neighbor discovery optimizations (RFC 6775) were also proposed. Considered a typical link layer for LLN, IEEE 802.15.4 standard was the main reference. While other link layer standards are being considered nowadays, such as the Bluetooth Low Energy (BLE), their limited bandwidth and frame sizes still pose a challenge to the transmission of large packets. In these scenarios, it is also necessary to define routing mechanisms. RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks - RFC 6550) is the protocol proposed to address such needs.

Software-defined networking (SDN) is an approach to manage networks that separates the control plane from the data plane. The routing/forwarding decisions are made by the SDN controller based on network information received. Openflow [24] was the first protocol proposed to establish the communication between forwarding nodes and the SDN controller. Software Defined Wireless Sensor Networks (SDWSN) approaches are also presented in the literature [18], and we highlight SDN-Wise and TinySDN.

Despite all this proposals, WSN has not been widely deployed or used. We believe this has three main reasons: (1) challenges concerning resource constrains; (2) lack of integration of mechanisms proposed (MAC layer, routing, programming, management, security); (3) dedicated infrastructure for each WSN application.

Security for IoT and WSN is a major concern nowadays. These resource constrained devices usually do not implement security mechanisms to protect themselves and the data they collect and exchange. The hyper-connectivity of IoT is currently threatened by malicious and hostile entities that could perform remote attacks. Security incidents whose target is the IoT ecosystem are daily reported. The *US-CERT Alert (TA16-288A) - Heightened DDoS Threat Posed by Mirai and Other Botnets*, which was originally released on October 2016, describes how IoT devices were used to create large-scale botnets used to execute denial-of-service (DDoS) attacks. Therefore, security mechanisms with secure implementations should be carefully designed and deployed for IoT devices and networks.

Despite the problems faced by IoT devices recently, research on security algorithms for WSN and IoT has been discussed. Symmetric encryption algorithms can be executed in resource constrained devices [23, 31] without incurring in much overhead. iSMQV [16] and AdC [26] are proposals that address key exchange for IoT and constrained devices. Flauzac *et al.* [9] argue that security could be supported through SDN. Since the SDN controller knows the network topology, this information could be used to support node admission decisions, using an approach similar to the capacity sharing proposal [29]. Once nodes are admitted to an SDN domain, a key exchange or distribution mechanism could be used to support security services, such as data confidentiality, integrity and authenticity.

Our approach to increasing WSN usage is to coordinate all these efforts to enable the use of WSN infrastructure by multiple different applications and non technical users, implementing *Sensing as a Service*. To achieve this, we propose an SDN-based framework that includes communication protocols, node task programming, resource management and security services.

Thus, the main contribution in this paper is the design of a secure SDN-based framework. While some of the building blocks for our framework have already been designed and developed, the integration of such blocks is new, as well as the addition of the security mechanisms. Our framework relies on IT-SDN [1] to achieve SDWSN functionalities. It should provide confidentiality, authenticity and integrity to the control messages being exchanged. Furthermore, only authenticated nodes should have access to the communication infrastructure. Also, the algorithms selected should be quantum-resistant and should be securely implemented.

Notice that the term *Sensing as a Service* was also used by Sheng *et al.* [30]. In their work, they consider sensors on, or attached to, mobile phones collecting data through a cloud computing system. The focus on their vision paper is the requirements and challenges to implement such systems. Since they focus on mobile phones, most of their suggested future research directions are not applicable to WSN scenarios. Furthermore, security issues are not directly addressed. However, we seek common goals such as energy efficiency and a general reusable framework.

This paper is organized as follows. Section 2 reviews background and related work. Our framework to support secure *Sensing as a Service* is presented in Section 3, and its implementation is discussed in Section 4. Section 5 presents some use case scenarios. Lastly, Section 6 presents final considerations and future work.

2 Related Work

This section covers two main aspects: Software-Defined WSN and security approaches for WSN.

2.1 The SDN Approach in WSN

Applying the SDN (Software-Defined Networking) paradigm to WSN (Wireless Sensor Networks) imposes different challenges and requirements when compared to wired networks. The limited resources available on the devices (i.e., energy, processing, memory, bandwidth and payload size) are related to the main challenges. The requirements are related to the applications characteristics (e.g. sensing rate, data size, data transmission pattern and aggregation), as well to the nodes behavior due to duty-cycles, operating systems and programming approach.

On the other hand, opportunities provided by SDWSN (Software Defined Wireless Sensor Networks) include: to improve resource reuse, to implement node retasking, to establish and improve node and network management, as well as to enable experiments with new protocols and to ease transition to standard protocols for deployed networks [7]. Notice that these features are key to achieve the *Sensing as a Service* paradigm [22]. Several SDWSN approaches are presented in the literature [18]. The ones based on OpenFlow [20, 21] have issues concerning frame sizes, overhead introduced and the use of TCP as underlying communication protocol.

TinySDN [8] is a flow-ID-based approach that improves on previous work by enabling the use of multiple SDN controllers and by addressing (i) in-band control (i.e., control and data packets share the same bandwidth); (ii) higher communication latency; (iii) smaller link layer frames; and (iv) limited energy supply. TinySDN provides the code and related documentation, but it is highly dependent on TinyOS [13], thus limiting the platforms it could be deployed. Furthermore the use of the ActiveMessage function limits interoperability with other systems.

SDN-WISE [10] defines mechanisms for a stateful Flow Table, pursuing two goals: (i) to reduce the amount of information exchanged between sensor nodes and the SDN network controller, and (ii) to make sensor nodes programmable as finite state machines. SDN-Wise implementation is not completely available for download and use. Given the limitations of the approaches available on the literature, we recently introduced IT-SDN [1]. IT-SDN is an SDWSN tool that is completely open and available, unlike SDN-Wise. While its design is inspired by TinySDN [8], we improved the architecture, protocols and implementation.

None of the aforementioned SDWSN frameworks include security services for the control or data plane.

2.2 Security in WSN

Several WSNs applications require security services, such as confidentiality, integrity and authenticity. Given the resource constrained requirements of WSNs, several specific security architectures based on symmetric cryptography were proposed, such as TinySec [17], Minisec [19], ContikiSec [3], and WSNSec2012 [2]. Also the IEEE 802.15.4 standard includes operation modes to provide security services on the link layer [14]. All these proposals focus on hop-by-hop security, since they were designed/implemented to operate at the link and/or network layer. Unfortunately they cannot guarantee end-to-end security, which is addressed by WSN-ETESec [27]. Furthermore, most of these architectures were designed and implemented for specific hardware platforms and operating systems, preventing its wide adoption.

An important issue to the effectiveness of these architectures is how secret keys are distributed. Key establishment approaches for WSNs must satisfy several security and functional requirements, which are often conflicting, such as resilience, authentication, connectivity and reduced memory usage.

Solutions for key distribution in WSNs are classified into three categories: pre-distribution, arbitrated, and self-enforcing [33]. Pre-distribution keying approach involves loading keys into sensor nodes prior to deployment. Arbitrated protocols use more powerful nodes to execute complex tasks, while self-enforcing is the dynamic establishment of keys using, generally, asymmetric cryptography.

Authentication of Things (AoT) [26] is a suite of protocols that incorporate authentication and access control designed for IoT. Primarily, AoT relies on Identity- and Attribute-Based Cryptography to cryptographically enforce Attribute-Based Access Control (ABAC).

iSMQV [16] is a combination of SMQV (strengthened-Menezes-Qu-Vanstone) with implicit certificates, which leads to a secure, lightweight, and escrow-free authenticated key agreement (AKA) protocol. This protocol is used for bootstrapping keys between authorized nodes, avoiding the transmission of quite large certificates and the execution of memory- and processing-intensive cryptographic algorithms, which are not suitable for WSNs. Escrow-free is an important property, since key-escrow enables a fully trusted authority to know the private keys of all nodes.

IETF *Authentication and Authorization for*

Constrained Environments (ace) working group[1] proposed a standard for an architecture for authorization in constrained environments, and then proposed a standard named *Authentication and Authorization for Constrained Environments*, which uses as building blocks OAuth 2.0 (RFC6749) and CoAP (RFC 7252). Notice that these proposals focus on the application layer, and thus consider the use of the Transport Layer Security (TLS) protocol (RFC 5246). TLS considers the following key exchange algorithms: RSA, Digital Signature Standard (DSS - FIPS 186) and Elliptic Curve Digital Signature Algorithm (ECDSA). Therefore, these approaches are not comparable to the ones previously described, since they rely on certificates.

3 SSAAS FRAMEWORK

As discussed in the Introduction, the *Sensing as a Service* paradigm could increase WSN usage, since it would enable the use of WSN infrastructure by multiple different applications and non technical users. To achieve this, we propose a secure SDN-based framework that includes communication protocols, node task programming, resource management and security services. Next we present an overview of the *Secure Sensing as a Service* (SSaaS) framework, and its requirements.

3.1 Overview

An overview of the SSaaS framework is depicted in Figure 1.

The **SSaaS manager** orchestrates the *SDN controller*, the *application controller*, and the *security controller*, along with the resource and topology information obtained by them.

The *SDN controller* receives information from the SDN devices regarding neighbor information (according to the specified neighbor discovery protocol), and builds a global topology map. Based on this, it will be able to proactively set routes (or flows) or reply to route (or flow) requests from the SDN devices.

The *application controller* provides an interface to the user, so that he/she can select the WSN it would like to program, and the corresponding tasks. The task programming includes selecting: what sensors to use (e.g. temperature, light, accelerometer), sensing rate, data transmission rate, WSN sink address, duration of the task (e.g., once, one hour, one day, always). The application controller makes use of a middleware (such as WARM [32]) to send the information to the WSN node. The middleware is also responsible to collect

[1] `https://datatracker.ietf.org/wg/ace/documents/`

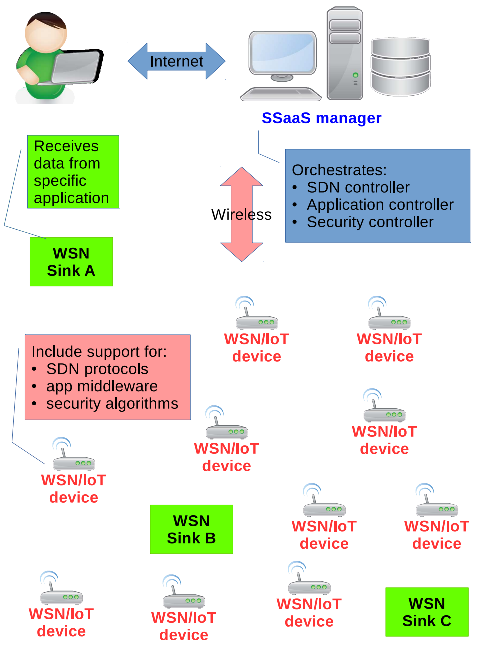


Figure 1: *Sensing as a Service* (SSaaS) overview

information about the WSN device resources (e.g., hardware configuration, communication capabilities, sensors available, tasks supported). The application controller must also keep track of the tasks allocated to each device.

The *security controller* encompasses two main functions: node admission, and key distribution (or generation, depending on the approach). Node admission is related to accepting a device to participate in the network, i.e., the device should be able to take part in southbound, neighbor discovery and controller discovery protocols, as described in Section 4.1. Another level of node admission could be related to the middleware used or a specific application using the WSN services.

Key distribution is necessary in order to establish a secure communication channel between nodes and controllers (i.e., to support the control plane), or between nodes and WSN sink, or between nodes (in both cases to support the data plane). The secure communication channel could provide confidentiality, integrity and/or authenticity to the message exchanged. This security can be achieved using symmetric algorithms (in which case both ends should share the same key), or asymmetric algorithms (in which case nodes must exchange public keys). The selection of the algorithms depends on the platform capabilities and the security level required (e.g., 128-bit security).

The **WSN devices** should include support for the

SDN protocols on the selected SDWSN framework, the application middleware, and security algorithms to achieve the security functionalities.

The **WSN sink** receives data from a specific application configured on the application server. It does not have to be controlled by the same entity that controls the SSaaS framework, but it needs to be admitted in the infrastructure. It also should be compatible with the midleware used by the *application controller*, since it will receive data collected.

3.2 Requirements

The SSaaS framework requirements are related to infrastructure and security characteristics. The infrastructure requirements are as follows.

- **Communication:** routing is achieved through an SDN-based approach, which relies on the *SDN controller* to: (i) receive neighbor information from the SDN-enabled devices, (ii) build a global topology map, and (iii) set routes (or flows) on the SDN-enabled devices.
- **Node programming and retasking:** a middleware able to address node task programming and retasking is necessary. The user interacts with the *application controller* to select the WSN nodes it would like to program and the corresponding tasks. Then, the *application controller* uses the middleware to configure the WSN nodes.
- **Resource database:** the SSaaS framework, by itself or by coordinating the SDN-based approach and the middleware databases, must store and manage information about: (i) the WSN device resources (e.g., hardware configuration, communication capabilities, sensors available, tasks supported); (ii) the tasks allocated to each device; (iii) the routes (or flows) set up; (iv) the existing WSN sinks. It is also relevant to obtain network and devices statistics, in order to balance resources usage.
- **WSN devices and sinks:** should support the protocols and mechanisms selected for the infrastructure.

We can organize the security requirements in three main categories:

- **Security services:** should include confidentiality, authenticity and integrity. These services should be available to the control messages being exchanged, as well as support application that requires such services. To achieve such services, we need the algorithms implemented on the devices. If symmetric algorithms are selected, which are usually lightweight and require less resources, than we need some approach to determine the shared keys.
- **Node admission:** depending on the infrastructure requirements, only authenticated nodes should have access to the communication infrastructure. Thus, node admission should enable the key exchange mechanisms. Furthermore, it is necessary to determine how a given node will be authenticated.
- **Long-term security:** could be achieved through post-quantum (i.e. quantum-resistant) algorithms. Thus, it is important to select such algorithms, which should be securely implemented.

3.3 Assumptions

We assume that the controllers in our system are trusted entities, i.e., they will not impersonate other devices and they will share symmetric keys with authorized nodes only. Therefore the implementation of the security mechanisms, as well as the protocols and softwares, must be secure so that it is not possible to compromise an authentic controller. Furthermore, messages and key exchange with controllers should make use of mechanisms to assure authenticity. Thus it would not be possible that an attacker impersonates a controller on the network.

We consider that the communication between nodes and controllers uses wireless media, and thus it could be eavesdropped. Node deployment may be planned by one entity and carried out by another entity. Node admission criteria (or access control policy) is out of the scope of this work, but it is relevant to determine which devices could take part in the communication and sensing infrastructure.

4 IMPLEMENTATION

In this section we discuss the approaches and mechanisms to comply with the requirements described previously. We introduce and discuss the SDN approach for WSN and IoT communication, followed by the WSN programming and network management. The security topics are discussed next.

4.1 SDWSN framework

IT-SDN [1] is an SDWSN (Software Defined Wireless Sensor Networks) tool that is completely open and available. Its design is inspired by TinySDN [8], but we improved the architecture, protocols and

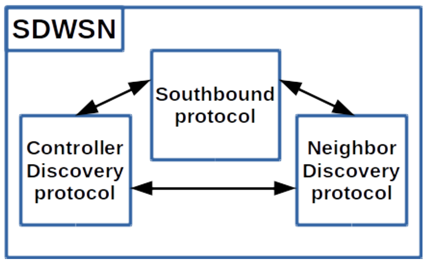


Figure 2: SDWSN architecture [1]

implementation. When designing IT-SDN, we consider that the SDWSN architecture contains three main communication protocols: southbound (SB), neighbor discovery (ND) and controller discovery (CD), as illustrated in Figure 2. The SB protocol is used for the communication between controller and SDN-enabled devices. It defines the packet format, how these packets are processed and the state machine. The ND protocol is used to obtain and maintain node neighborhood information. Lastly, the CD protocol identifies a next hop candidate to reach the controller.

IT-SDN underlying architecture is independent of the operating system and its functions, and this is key to enable interoperability. Also, IT-SDN defines a clear separation of the protocols used to achieve SB, ND, and CD, and this creates the environment to evaluate new protocols to achieve these tasks depending on the network characteristics. Furthermore, IT-SDN implementation supports configuring multiple nodes with a single packet, a feature that was envisioned in previous work but has not been implemented [8, 10].

Figure 3 depicts the message exchange between an SDWSN node and the controller. Once the controller knows its neighbors (i.e., it has received neighbor report messages), it has a partial topology view and is able to send flow setup messages. When a node receives a flow setup message, it will reply with a neighbor report, which will improve the controller network view. When a node receives a packet for which it does not have a matching rule on its flow table, it sends a flow setup request to the controller. After receiving the flow setup message, the node is be able to process such packet.

IT-SDN Performance Evaluation

In order to evaluate IT-SDN performance we used COOJA [28], a WSN node emulator with integrated radio medium simulation. We selected DGRM (Directed Graph Radio Medium) to control the network topology. We chose the TelosB device as the basis of our simulations. The scenario includes a controller running on a PC, which communicates with the emulated

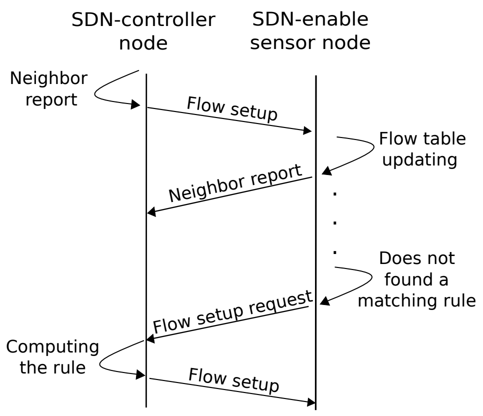


Figure 3: IT-SDN message exchange [1]

network through "Serial Server" COOJA plugin, and was positioned in the corner of the square grid topology. One node was set as the data sink, and regular nodes transmitted data towards the sink at 1 packet/min. The network size ranged from 9 to 25 nodes. The following metrics are assessed: (1) delivery, (2) delay, and (3) control overhead. Each scenario was simulated 20 times for 30 minutes, in order to obtain statistical significance.

Delivery rate is defined as the end-to-end delivery ratio. For the results depicted in Figure 4, we consider all the attempted transmissions, including message retransmissions for control messages. For instance, if a control message needs to be transmitted twice the delivery rate would be 50%. In this case, the delivery rate is near 100% for the network with 9 nodes, and decreases to around 65% in larger networks (25 nodes). Since the number of nodes generating data increases and the number of hops to be traversed increases as well, it is reasonable to see the decrease in the data delivery rate. Figure 5 depicts results for delivery rate excluding control message retransmissions. For larger networks, we notice an increase in the control message delivery rate.

Delay is defined as the time a packet takes to reach its final destination (including queue time, and route request-response delay). Control packets delay are not significantly affected by the increase in the network size, as illustrated in Figure 6. On the other hand, data packet delay increases with the network size, what is reasonable since the number of hops that must be traversed to reach the sink increases. Control overhead, depicted in Figure 7, is given by the total number of non-data packets transmitted per node per minute. A control packet retransmission is processed as a new packet transmission for metric calculation purposes.

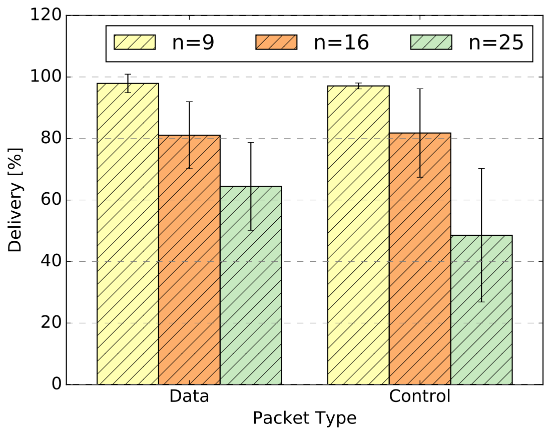


Figure 4: Delivery rate

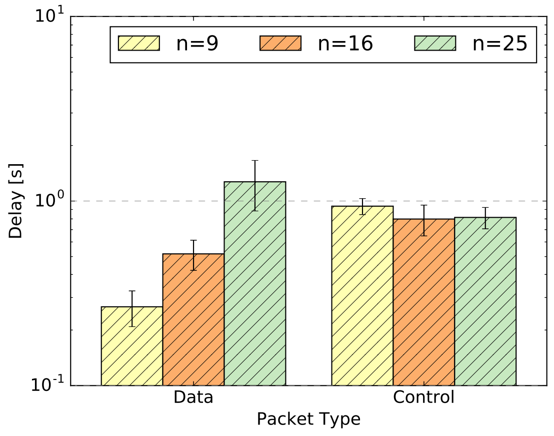


Figure 6: Delay

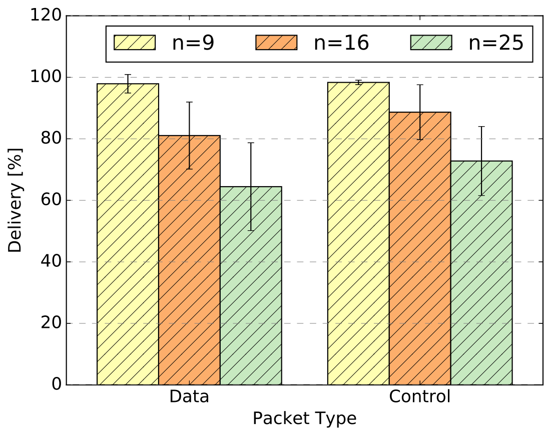


Figure 5: Delivery rate excluding control message retransmissions

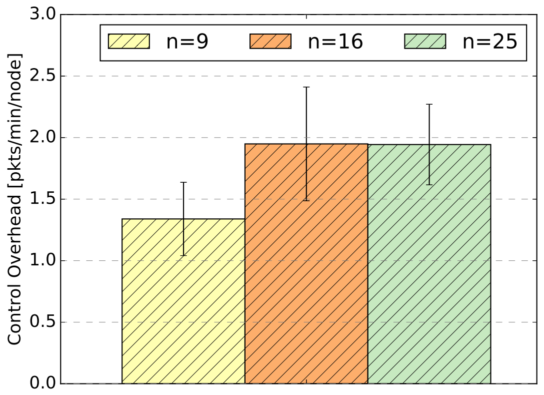


Figure 7: Overhead

4.2 WSN Programming and Management

WSN application development and resource management are still complex tasks. The diversity of resource constrained platforms and network protocols demands specialized knowledge and practice. Furthermore, once nodes are deployed software updates and node reprogramming depend on specific features already available on operating systems or software running on the devices.

We selected to use WARM (WSN Application development and Resource Management). WARM [32] is a framework that employs Web Service and Software Defined Networking paradigms to enable the parametrized scheduling of tasks. It allows the user to configure typical applications running on a simulated WSN using a web browser.

WARM is based on the concept of application tasks, which are hardware dependent routines whose instances can be scheduled in sensor nodes. These routines should implement lightweight processing algorithms, environmental sensing or actuator actions, each one representing a different capability of a sensor node. Furthermore, they can also make use of the provided SDN-based IEEE 802.15.4 stack in order to receive input data from other tasks or to send their output data as input for other tasks through the sensor network.

4.3 Security

To achieve the requirements identified, we need to employ security algorithms. Symmetric algorithms are more efficient in terms of resources necessary (i.e., processing, memory, energy, message sizes). However, in order to use such algorithms, we need to achieve key exchange or distribution, as discussed before (Section 2.2). To address the long-term

Table 1: Comparison of parameter sizes (in bytes) with various post-quantum signature schemes at the quantum 128-bit security level (Based on [34], with added information)

Scheme	Public key size	Private key size	Signature size
Hash-based	1056	1088	41000
Code-based	192192	1400288	370
Lattice			
NTRU MLS	886	831	50
BLISS	7168	2048	5120
Tesla#	7168	4608	3488
Multivariate	99100	74000	424
Isogeny	768	48	141312
Compressed	336	48	122880
ECC (non PQ)	32	32	

Table 2: Comparison of number of messages to transmit a signature using IEEE 802.15.4 payload and amount of memory to store public keys (in bytes) with various post-quantum signature schemes at the quantum 128-bit security level

Scheme	Number of Messages	Memory to store 10 Public Keys (in bytes)
Hash-based	410	10,560
Code-based	4	1,921,920
Lattice		
NTRU MLS	1	8,860
BLISS	52	71,680
Tesla#	35	71,680
Multivariate	5	991,000
Isogeny	1,414	7,680
Compressed	1,229	3,360

security requirement, we should select post-quantum cryptography to achieve key distribution or exchange [5].

Post-Quantum Security

Post-quantum cryptography refers to cryptographic algorithms that are thought to be secure against an attack by a quantum computer. The public-key algorithms can be organized in the following classes: (1) lattice-based (2) multivariate (3) hash-based (4) code-based and (5) supersingular elliptic curve isogeny cryptography.

While some authors argue that ECC algorithms provide the necessary security level for nowadays, we should consider long-term security since deployed algorithms are not be replaced easily. Furthermore, quantum computers are not far from reality given IBM's and Google's recent announcements[2]. Also the National Institute of Standards and Technology (NIST) has published a report on the subject [4], and is now accepting submissions for quantum-resistant public-key cryptographic algorithms. Despite the threat of quantum computers, the integration of the quantum-resistant cryptographic algorithms (i.e., the so called post-quantum cryptography) [5] in the context of WSN has not been widely discussed before.

Post-quantum protection can be employed in different security operations, such as encryption, decryption and signature schemes. Yoo et al. [34] provide a comparison of parameter sizes (in bytes) with different post-quantum signature schemes at the quantum 128-bit security level, which is depicted in Table 1. The post-quantum signature schemes are: stateless hash-based signature SPHINCS-256, a code-based signature based on Niederreiters variant of the McEliece cryptosystem, a lattice-based signature BLISS, a ring-LWE-based signature TESLA#, the multivariate polynomial-based Rainbow signature, and the isogeny-based scheme and its compressed version proposed [34]. We also included in the table the results we obtained for NTRU MLS, a lattice-based signature scheme, identified by the set of parameters xxx-20151024-443 [3], as well as the non post-quantum algorithm ECC for reference purposes.

Given resource constrained devices we consider for WSN and IoT, it is important that the algorithms require a low memory footprint to store the keys and a small amount of data to be transmitted (public keys and signatures). Typical WSN constrained devices are characterized by small memory capacity, such as 10 KB of RAM and 48 KB of programmable flash memory of the TelosB [25]. Thus, key sizes of kilobytes are not feasible for such devices, since, besides storing their own pair of keys (public and private), nodes should store the public keys of nodes they must exchange information with.

The IEEE 802.15.4 standard [14] has a 127-byte frame size, which must include the its own headers as well as the information from upper layers. Thus a signature from a code-based scheme is attractive, since it would require about 4 messages to be transmitted. On the other hand, transmitting a code-based public key would require about 1,900 messages, thus making this scheme unsuitable to such environments. Therefore it is important to select an algorithm whose public key and signature sizes provide a good balance in terms of amount of messages transmitted.

Table 2 depicts the number of messages to transmit

[2] MIT Technology Review written by Tom Simonite on April 21, 2017. Available at `https://goo.gl/DIyB4X`.

[3] Results from EISEC/TUM implementation.

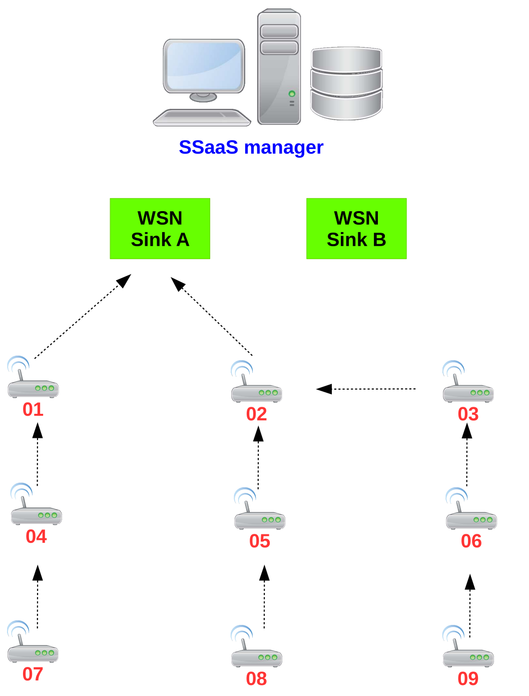


Figure 8: Infrastructure usage with one sink

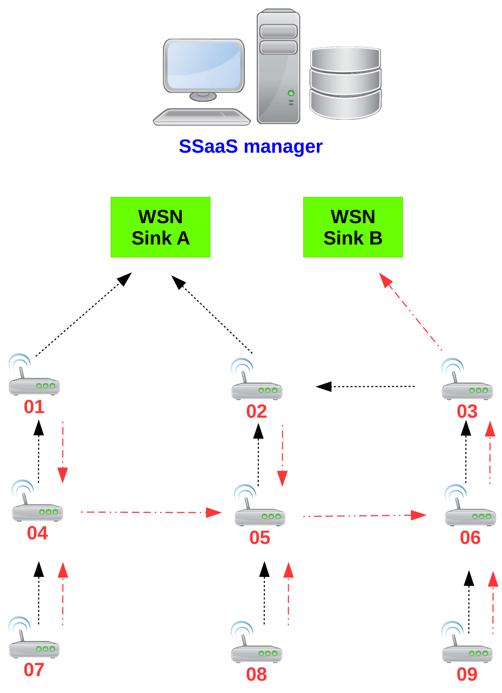


Figure 9: Infrastructure sharing with different sinks

a signature using the IEEE 802.15.4 payload (around 100 bytes), as well as the amount of memory necessary to store ten public keys (in bytes) with the post-quantum signature schemes depicted in Table 1. For this comparison, we considered that the devices would need to store 10 public keys, which would be the public key from the controllers and other nodes with which they would exchange data. Therefore, NTRU MLS is the best option in terms of post-quantum algorithm, since it achieves the best results in terms of number of messages to be transmitted and amount of memory to store keys.

Another aspect that should be considered is the execution time of such algorithms in resource constrained platforms. For instance, considering 128-bit security level, NTRU Encryption takes 588,044 cycles to run in Cortex M0, while decryption takes 950,371 cycles [12]. Since the Cortex M0 runs at 32 MHz, it would take about 18 ms to encrypt and 30 ms to decrypt, which is comparable to the time it takes to obtain a temperature reading on the SensorTag device (25 ms) [31].

5 SSaaS Use Case

Usually, WSN nodes are considered disposable and cheap devices, which could be deployed for a specific task [7]. This is not the case in a smart city scenario, where sensor nodes should collect, process and transmit different types of data for different applications (e.g. environmental monitoring, traffic monitoring and control, surveillance). If these sensor nodes and other devices collecting data could be managed by the SSaaS framework, one could achieve a much better usage of the underlying infrastructure.

The SSaaS framework benefits from the SDN paradigm centralized view of the network, which enables node and resource management. Furthermore, the SSaaS manager could use the energy available in a given node (or set of nodes) or the security trust to determine different weights on the topology graph, and make the *SDN controller* select routes that will provide the best network lifetime or the most secure path. Based on information from the *application controller* concerning the WSN application requirements, the SSaaS manager could make the *SDN controller* select different parameters to determine the route used by each application flow.

Furthermore, the SSaaS manager could request the *application controller* to relocate certain tasks, in order to process more user's requests. For instance, consider the following situation. Application A is collecting data and sending it to WSN sink A, as depicted in Figure 8. Then application B is instantiated to collect and send data to WSN sink B. If routes are reused, this could lead to node 02 energy depletion before other nodes. Thus it is better to create different routes, in order to balance the energy consumption, as shown in Figure 9.

6 Conclusion

In this paper, we presented a secure SDN-based framework, *Secure Sensing as a Service* (SSaaS) framework, which includes: communication protocols, node task programming middleware, communication and computation resource management features and security services. The communication layer for the constrained devices considers IT-SDN as its basis. We include novel performance evaluation results from IT-SDN, assessing: delivery rate, delay, and control overhead.

Concerning security, we address the main services, the type of algorithms to achieve them, and how their secure implementation is needed to avoid attacks. Furthermore, we discuss how to employ post-quantum algorithms in our framework.

We are currently working on the integration of the components, and next we will evaluate its overall performance. In the future, we plan to execute the SSaaS framework on our testbed at the USP, with nodes deployed throughout the building.

Acknowledgment

C. B. Margi would like to thank Prof. Dr.-Ing. Georg Sigl, from EISEC/TUM, for the discussions and hosting her as visiting researcher (FAPESP fellowship #2016/06930-8). The authors would like to thank Bruno T. de Oliveira for his collaboration on the early stages of this research. This project was partially funded by FAPESP-Intel research grant #2015/50520-6. C. B. Margi is supported by CNPq fellowship (Pq) #307304/2015-9. R. C. A. Alves is supported by CNPq PhD fellowship #155372/2016-5.

References

[1] R. C. A. Alves, D. A. G. Oliveira, G. A. Nez, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," in *SBRC 2017 - Salo de Ferramentas*, May 2017. [Online]. Available: http://www.larc.usp.br/~cbmargi/it-sdn/it-sdn-sbrc2017sf.pdf

[2] N. Bandirmali and I. Erturk, "WSNSec: A scalable data link layer security protocol for WSNs," *Ad Hoc Networks*, vol. 10, no. 1, pp. 37–45, 2012.

[3] L. Casado and P. Tsigas, "ContikiSec: A Secure Network Layer for Wireless Sensor Networks Under the Contiki Operating System," in *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*, ser. NordSec '09, Berlin, Heidelberg, 2009, pp. 133–147.

[4] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone.

[5] C. Cheng, R. Lu, A. Petzoldt, and T. Takagi, "Securing the Internet of Things in a Quantum World," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 116–120, February 2017.

[6] D. Culler, D. Estrin, and M. Srivastava, "Overview of Sensor Networks," *Computer Magazine*, vol. 37, no. 8, pp. 41–49, 2004.

[7] B. T. de Oliveira, R. C. A. Alves, and C. B. Margi, "Software-defined wireless sensor networks and internet of things standardization synergism," in *Standards for Communications and Networking (CSCN), 2015 IEEE Conference on*, October 2015, pp. 96–101.

[8] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Communications (LATINCOM), 2014 IEEE Latin-America Conference on*, Nov 2014, pp. 1–6.

[9] O. Flauzac, C. Gonzalez, A. Hachani, and F. Nolot, "SDN Based Architecture for IoT and Improvement of the Security," in *Advanced Information Networking and Applications Workshops (WAINA), 29th International Conference on*, March 2015, pp. 688–693.

[10] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE : Design , prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," *IEEE Conference on Computer Communications (INFOCOM)*, pp. 513–521, 2015.

[11] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09, New York, NY, USA, 2009.

[12] O. M. Guillen, T. Pppelmann, J. M. Bermudo Mera, E. Fuentes Bongenaar, G. Sigl, and J. Sepulveda, ""towards post-quantum security for iot endpoints with ntru"," in *Design Automation and Test in Europe (DATE 2017)*, Lausanne, Switzerland, Mar 2017.

[13] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, 2000.

[14] IEEE, "802.15.4-2011 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," 2011.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '00, New York, NY, USA, 2000, pp. 56–67.

[16] M. A. S. Jr., M. V. Silva, R. C. Alves, and T. K. Shibata, ""lightweight and escrow-less authenticated key agreement for the internet of things "," *Computer Communications*, vol. 98, pp. 43 – 51, 2017.

[17] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, New York, NY, USA, 2004, pp. 162–175.

[18] H. I. Kobo, A. M. Mahfouz, and G. P. Hancke, "A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements," *IEEE Access*, vol. 5, pp. 1872–1899, 2017.

[19] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: a secure sensor network communication architecture," in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, New York, NY, USA, 2007, pp. 479–488.

[20] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[21] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, vol. 1, 2011, pp. 594–600.

[22] C. B. Margi, "Comunicação, segurança e gerenciamento em redes de sensores sem fio," 2015, tese (Livre-Docência) – Universidade de São Paulo.

[23] C. B. Margi, B. T. de Oliveira, G. T. de Sousa, M. A. Simplício, P. S. L. M. Barreto, T. C. M. B. Carvalho, M. Näslund, and R. Gold, "Impact of operating systems on wireless sensor networks (security) applications and testbeds," in *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, Aug. 2010, pp. 1–6.

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[25] MEMSIC Inc., "TelosB Product Details," http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf, Massachusetts, USA, 2012, 6020-0094-04 Rev B.

[26] A. L. M. Neto, A. L. F. Souza, I. Cunha, M. Nogueira, I. O. Nunes, L. Cotta, N. Gentille, A. A. F. Loureiro, D. F. Aranha, H. K. Patil, and L. B. Oliveira, "AoT: Authentication and Access Control for the Entire IoT Device Life-Cycle," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, ser. SenSys '16, New York, NY, USA, 2016, pp. 1–15.

[27] B. T. Oliveira and C. B. Margi, "WSN-ETESec: Criptografia fim-a-fim em redes de sensores sem fio," in *Anais do XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - Trilha Principal e Salão de Ferramentas*. Porto Alegre - RS: Sociedade Brasileira de Computação, 2012, pp. 930–937.

[28] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Local Computer Networks, Proceedings 31st IEEE Conference on*, Nov 2006, pp. 641–648.

[29] M. A. S. Santos, B. T. de Oliveira, C. B. Margi, B. Nunes, T. Turletti, and K. Obraczka, "Software-defined networking based capacity sharing in hybrid networks," in *Proceedings of Capacity Sharing Workshop (CSWS'13), In conjunction with ICNP'13*, May 2013.

[30] X. Sheng, J. Tang, X. Xiao, and G. Xue, "Sensing as a Service: Challenges, Solutions and Future Directions," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3733–3741, Oct 2013.

[31] T. Shibata, R. de Azevedo, B. C. Albertini, and C. B. Margi, "Energy consumption and execution time characterization for the SensorTag IoT platform," in *XXXIV Simpsio Brasileiro de Telecomunicaes e Processamento de Sinais (SBrT 2016)*, Santarém, Brazil, Aug. 2016, pp. 55–59.

[32] H. Silva, A. H. Pereira, Y. Solano, B. T. de Oliveira, and C. B. Margi, "WARM: WSN application development and resource management," in *XXXIV Simpsio Brasileiro de Telecomunicaes e Processamento de Sinais*. Santarm, Brazil: Sociedade Brasileira de Telecomunicaes, 2016.

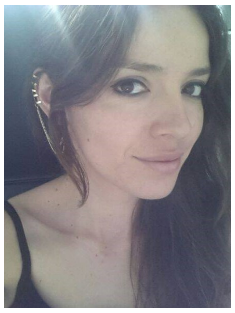
[33] M. A. Simplício, P. S. Barreto, C. B. Margi, and T. C. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks*, vol. 54, no. 15, pp. 2591–2612, 2010.

[34] Y. Yoo, R. Azarderakhsh, A. Jalali, D. Jao, and V. Soukharev, "A Post-Quantum Digital Signature Scheme Based on Supersingular Isogenies," Cryptology ePrint Archive, Report 2017/186, 2017, http://eprint.iacr.org/2017/186.

AUTHOR BIOGRAPHIES

Cintia Borges Margi is Associate Professor at Universidade de São Paulo since 2015, where she started as Assistant Professor in February 2007. She obtained her degree in Electrical Engineering (1997) and her MSc in Electrical Engineering (2000) at Universidade of São Paulo, her PhD in Computer Engineering at University of California Santa Cruz (2006), and her Habilitation (Livre Docência) (2015) in Computer Networks from the Universidade of São Paulo. Her research interests include: wireless sensor networks (protocols, systems, security, energy consumption and management, embedded hardware) and software-defined networking.

Renan C. A. Alves is a PhD student at University of São Paulo. He obtained his M.Sc degree (2014) and graduated in Electrical Engineering with emphasis in Computer and Digital Systems (2011) both from University of São Paulo. His main research interests include network protocol modeling and performance analysis, Wireless Sensor Networks protocols and applications.



Johanna Sepulveda received the M.Sc. and Ph.D. degrees in Electrical Engineering - Microelectronics by the University of São Paulo, Brazil in 2006 and 2011, respectively. She was Post-doctoral fellow at the Integrated Systems and Embedded Software group at this University and at the group of Embedded Security of the University of South Brittany, France. Moreover, Dr. Sepulveda was a Visiting Researcher at the Computer Architecture group at the University of Bremen, Germany. In 2014, she worked as a Senior INRIA Postdoctoral researcher at the Heterogeneous Systems group at the University of Lyon, France. Since 2015, she holds a Senior Researcher Assistant position at the Technical University of Munich, Germany. She has been working in the field of embedded security design for more than 10 years. Her research interest also includes high performance SoC design and protected post-quantum security deployment.