Enabling security in software-defined wireless sensor networks for internet of things

Cézar M. G. de Toledo, Doriedson A. G. de Oliveira, Marcos A. Simplicio Jr and Cintia B. Margi

[1]Laboratório de Arquitetura e Redes de Computadores – Universidade de São Paulo (USP)
CEP – 05508-010 – São Paulo – SP – Brazil

{cmurilo,doliveira,msimplicio,cbmargi}@larc.usp.br

Abstract. *In Software-Defined Wireless Sensor Networks (SDWSN), a logically centralized controller manages data flows according to high level policies. As a result, it provides Wireless Sensor Networks (WSN) with further flexibility and control over its nodes' behavior. One challenge in this scenario, though, is that SDWSN are mainly composed of resource-constrained devices, which hinders the application of traditional cryptographic protocols in such networks. In this article, we propose a secure framework for SDWSN that takes into account such constraints, enabling the establishment of end-to-end security among nodes and between nodes and the SDN controller. Besides showing how our proposal can enforce different security services in an SDWSN, we also simulate our framework and present a preliminary security cost of framework.*

1. Introduction

In traditional networks, the switches' control guidelines are pre-installed, and play a critical role on the correct and efficient processing of packets. As a drawback of this approach, managing the switches usually requires a lot of effort from network administrators. After all, every single node needs to be (maybe manually) configured to ensure that interactions between them lead to the expected results.

Software Defined Networking (SDN) promises to ease this management issue by creating a network logic that is (1) decoupled from the underlying hardware and (2) managed by a remote control plane. More precisely, the SDN paradigm considers three planes that are logically connected: data, control, and application planes. The data plane comprises the switches, whose only function is forwarding packets. The control plane consists in the controller, which manages all network logic by defining how packets should be handled by each switch. And the application plane is where users and administrators interact with the controller, developing applications and managing the state of the network.

The SDN paradigm has become more prevalent thanks to the Southbound-based OpenFlow protocol, which is currently supported by many network deployments. One potential limitation of OpenFlow (OF), though, is that its design is somewhat oriented toward wired networks [10]. In particular, it assumes high-speed switches that (1) can contact the network controller as often as necessary, and (2) can store (possibly large) tables with flow rules. Hence, OpenFlow does not always cope with constraints commonly found in Wireless Sensor Networks (WSNs), namely: (1) prevalence of low speed network protocols, like IEEE 802.15.4; and (2) limited processing power and memory

availability on the network nodes, which act simultaneously as end nodes and switches. Indeed, these challenges have motivated many recent research efforts focused on creating a Software-Defined WSN (SDWSN) environment [1, 5]. The goal of SDWSNs is, thus, to enable dynamic and scalable routing for accommodating the different needs of Internet-of-Things (IoT) applications on the same WSN. For example, the SDN paradigm can help WSNs to balance and/or optimize multiple metrics, such as energy, latency or reliability, depending on current network usage and policies.

Albeit promising, SDWSNs are prone to security vulnerabilities inherited from SDNs, such as link spoofing [2, 13], access control abuse [7], denial of service (DoS) [9], and man-in-the-middle (MitM) [4]. At the same time, they are also prone to WSN-related attacks, including sink holes [8] and additional forms of DoS [14] and MitM [8]. Therefore, the widespread adoption of SDWSN technologies depends, at least in part, on a robust security framework for addressing such threats.

In this work, we address this issue by proposing a security framework that addresses some of the main threats found in the literature. Namely, by using adequate cryptographic mechanisms, our proposal can mitigate attacks involving the insertion of false sensors, controllers or sinks in the network.

This work is organized as follows. Sec. 2 describes the method adopted in this research. Sec. 3 describes the proposed security framework, while Sec. 4 evaluates its computational costs. Sec. 5 concludes the discussion.

2. Method

Most of the security issues mentioned in Section 1 stem from the absence of robust cryptographic protocols in existing SDWSN architectures. To address this issue, we make use of iSMQV [15], a lightweight authenticated key agreement (AKA) protocol designed specifically for WSNs. Basically, it comprises two mechanisms. The first is a security bootstrap protocol that enables nodes to get implicitly certified public-private key pairs, avoiding the transmission of large certificates commonly used in traditional networks [3]. These keys then enable any node to establish secret and authenticated key pairs with other nodes. In addition, iSMQV is escrow-free, i.e., the Key Generation Center (KGC) that authorizes nodes in the network does not learn their private keys. With the key pairs generated, message authenticity, integrity and confidentiality are provided by combining different symmetric cryptographic primitives.

Such mechanisms are then integrated into IT-SDN, an open-source SDWSN framework [12]. Even though other SDWSN exist [5, 11], this choice is motivated by IT-SDN's scalability, making it useful for large-scale deployments. Such scalability property comes mainly from the adoption of a source-routed approach for control packets, which reduces overheads and flow table occupancy. Indeed, IT-SDN can achieve a delivery rate of almost 100% of the control packets for a network composed of 64 nodes [1], and is also expected to scale for larger networks.

The resulting architecture assumes that all sensor nodes are SDN-enabled, so they can behave like switches, and that there is at least one controller among the nodes. The packet forwarding to a sink is done according to their address, which is advertised to the network controller when the sink enters the network. Controller and sink may run on different devices, and the network can accommodate multiple sinks.

3. Specification

The following requirements were taken into account in the design of the proposed security framework.

- *On-Demand Security*: Some nodes in the network may handle more sensitive data than others, so their need for authenticity, integrity and confidentiality may differ. Hence, the framework does not require that all nodes have valid keys from the start, which means that any sensor node that supports IT-SDN may be introduced in the network. Security services are then only employed if requested by a node, as long as the communication end-points have valid public-private key pairs.
- *Security Levels*: Some deployments may require a modern, 128-bit security level, while others may prefer a legacy, 80-bit security for compatibility purposes or to avoid some computational overhead. For this reason, the proposed framework supports both 128- and 80-bit security.
- *Support for different security services*: The network administrator can specify what security services are required for a flow. Three options are available: (0): No security service is required, so the communication avoids unnecessary overhead related to security algorithms; (1): Authentication-only, meaning that the packets' authenticity and integrity is ensured during the communication using a message authentication code (MAC); (2): Authenticated-encryption, in which case an authenticated-encryption with associated data (AEAD) scheme is employed to provide data confidentiality besides authenticity and integrity.
- *End-to-End Security*: Nodes support end-to-end secure communications between the controller and sensor nodes, as well as between sink and sensor nodes.

Such mechanisms, when combined, can be used to mitigate falsified sensor, controller and sink node [16]. Due to limited space we do not described such attacks here, since proposing and evaluating the framework are the main contribution here.

The control plane is responsible for defining the security services needed in the communication between controller and sensor nodes. The application plane is responsible for defining these services in the communication between sensor and sink nodes.

In the next subsections, we describe how such security services are integrated into the IT-SDN 0.4.1. We focus on how the proposed security services are built on top of IT-SDN, considering its capabilities.

3.1. Public/Private Key Bootstrap

The first step for any node that needs to engage in secure communications is to obtain a valid public-private key pair with an authorized Key Generation Center (KGC). In the proposed framework, we assume that this procedure is performed by the network administrator, in an out-of-band manner, using the iSMQV protocol [15]. As a result, the node obtains a private key r and a corresponding public key Y that is implicitly certified by the KGC. The adoption of implicit certification reduces the amount of information that needs to be exchanged by nodes when verifying the authenticity of each other's public keys. Namely, implicitly certified public keys consist of two elliptic curve points, which translate roughly to $4k$ bits of information for a system whose security level is k bits.

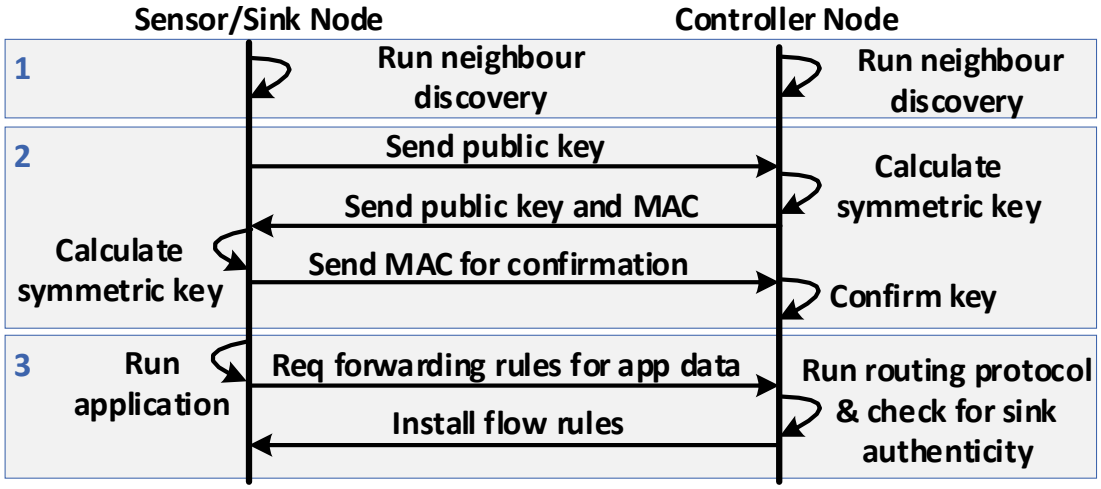

Figure 1. Key establishment with the controller

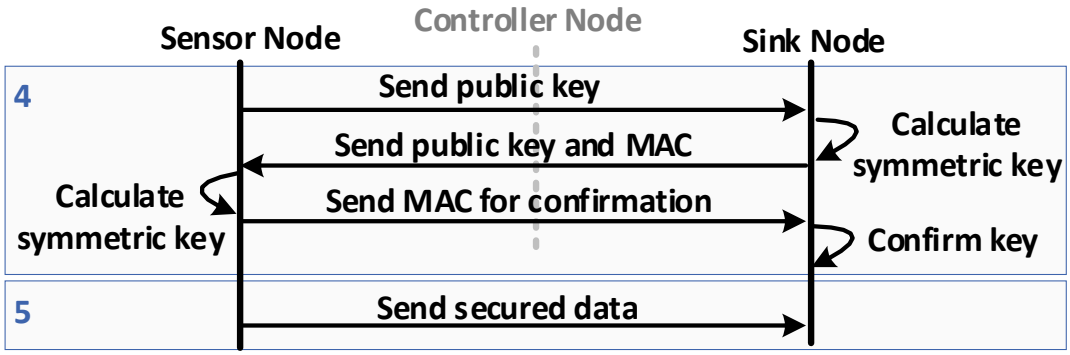

Figure 2. Key establishment between sensor and sink nodes

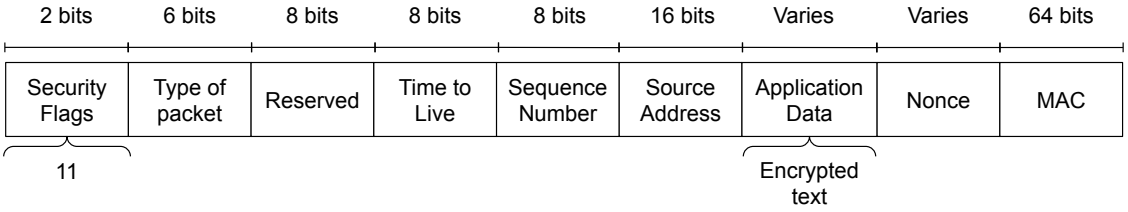

Figure 3. The header used in our security approach.

3.2. Symmetric Key Establishment

After entering the network, nodes that desire to use security services must establish a secure channel with the controller and with relevant sinks. This procedure is depicted in Figures 1 and 2, and can be summarized as follows:

1. First, sensors and sink nodes run the neighbour discovery protocol, learning about other nodes in the network and get a valid route to the controller.
2. All interested nodes then run the iSMQV protocol with the controller, aiming to establish a secure channel with it. This procedure requires nodes to inform their own public keys to the controller, which is combined with the controller's private key to generate a secret, symmetric key. The controller then responds with its own public key to the requesting node. Also, to confirm that a valid symmetric key was

calculated, the controller authenticates this response packet with the symmetric key, using a MAC scheme. Finally, the sensor/sink receives and generates the same key and sends an authenticated confirmation to the controller, proving that the protocol was correctly executed.

3. Subsequently, the sensor nodes can run applications and send data flow requests to the controller. If the application packet requires some security service, this need is also informed in the flow request. Unsecured requests are handled as in IT-SDN, by finding an appropriate sink and installing the required forwarding rules in the network. If some security service is required, the list of candidate sinks is restricted to authenticated nodes, i.e., sinks that have already established a secure channel with the controller; if no suitable candidate is found, an error message is sent to the requesting node.
4. When an authenticated sink is identified, the controller installs flow rules in the nodes, the sensor nodes can now establish a symmetric key with the sink node. The result is that an end-to-end secure channel is established between those nodes.
5. The application plane can enforce security services and send data to the sink.

3.3. Message format

To enforce the supported security services, we use 2 bits on the header of the IT-SDN packets, as depicted in Fig. 3. When a node receives messages in the system, it analyzes the security flags and take one of the following actions: (`0`) process the packet normally without any security services; (`1`) calculate the packet MAC to check its authenticity; (`2`) employ AEAD to check for its authenticity and decrypt the ciphertext.

Besides using such flag bits, the main modification made to the IT-SDN packets is the addition of a 64-bits MAC field for authentication, and the insertion of a nonce alongside the application data whenever an AEAD scheme is employed.

4. Benchmark

We simulate different scenarios comprising of 9, 25 and 36 nodes in a grid scheme. There was no further motivation to this choice, then the fact that we wanted to start with a small network and gradually increase it. First, we simulate a scenario of only one controller node, while the other scenario has one controller and one sink node. We focused our analyses on the key establishment protocol, which is the most costly procedure in the security framework. We use the Cooja simulator to evaluate our framework, the sensor nodes are considered to be the MSP430X microprocessor [6]. While the SDN controller was running on Linux, which for our work means the capabilities of the controller is far greater than the sensor nodes. We use the relic toolkit library to provide the cryptographic protocols, namely, we use the curves SECP160r1 and SECP256k1, which provide the 80- and 128-bit security levels supported in our framework. One thing to note is that if the security level is of 128 bits, it is necessary to send two messages with the public key. That is because of the IEEE 802.15.4 which limits the physical layer packets to 127 bytes. We use SHA-1 as the hash function used in the framework if the security level is set for 80 bits. While SHA-256 is the hash function used if the security level is set for 128 bits. We ran each scenario five times and collected the results for the first 20 minutes of simulation.

Fig. 4 shows how many nodes completed the authenticated key agreement with the controller and sink nodes. Our framework configured for 128 bits takes more time to

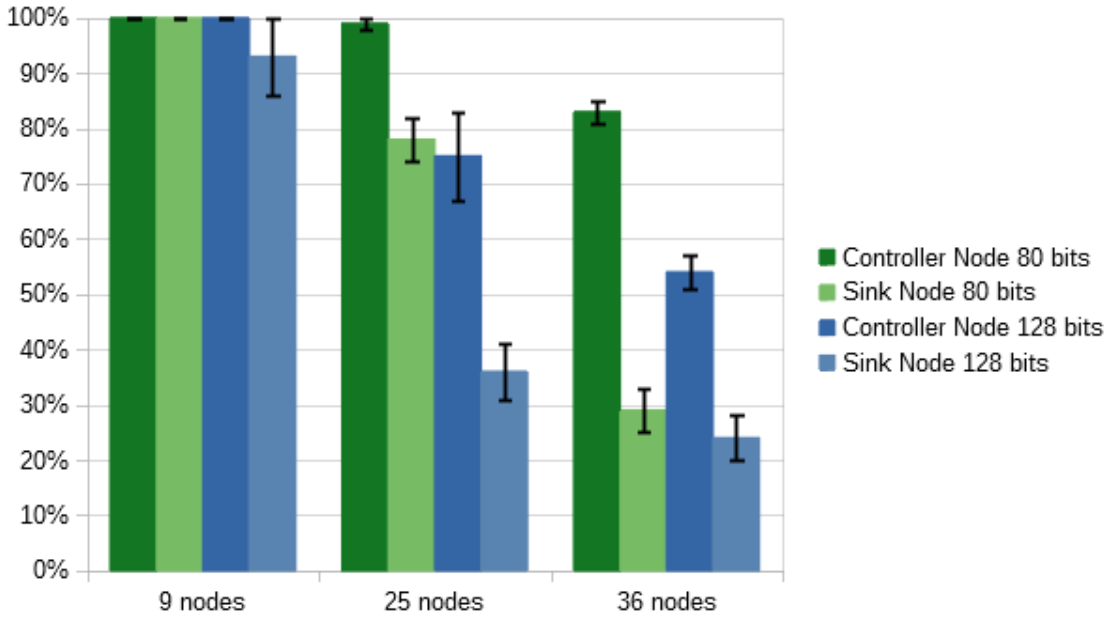

Figure 4. Percentage of nodes that completed the AKA process during the simulation time for each scenario.

establish the key with the nodes, mostly due to two factors; (1): there are two additional packets to establish the symmetric key and (2): the time to execute the iSMQV protocol is roughly 7 times slower in 128 bits. In addition we identified that the 80 bits/36 nodes scenario showed almost the same completion rate of the 128 bits scenario. This was due to packet loss. Packet collisions were common, and when the sink node is running the iSMQV protocol it becomes unreachable (since the controller node does not run on the MSP430X device, it does not suffer from this characteristic, hence, the results from the controller are far better). So, if a packet loss occurs, it is the IT-SDN re-transmission protocol responsible to send the packet again which might delay the entire process. We also inferred that larger networks poses memory problems with the sink node, it is not able to store keys from all nodes in the network. It is recommended the deployment of multiple sink nodes in this case. Finally, after completing the initial key agreement, there was no bottleneck identified by the other security standards introduced.

5. Conclusion

The goal of a Software-Defined Wireless Sensor Network (SDWSN) is to bring the flexibility provided by the SDN paradigm to the context of WSNs. Aiming to facilitate the deployment of security services in SDWSNs, in this article we describe a security framework designed specifically for such environments. The proposed solution builds upon lightweight cryptographic mechanisms to provide a variety of end-to-end security services for sensor nodes, sinks and controllers. It is particularly effective against falsification and eavesdropping attacks, as it enables authorized nodes to encrypt and authenticate packets whenever required. However, these mechanisms impose significant drawbacks during the start-up of the network. By enabling on-demand security we want to reduce this drawback, so that a more traditional network security model can be enforced on SDWSN.

References

[1] R. Alves, D. Oliveira, G. Segura, and C. Margi. The cost of software-defining things: A scalability study of software-defined sensor networks. *IEEE Access*, 7:115093–115108, 2019.

[2] A. Azzouni, R. Boutaba, N. T. M. Trang, and G. Pujolle. sOFTDP: Secure and efficient topology discovery protocol for SDN. *arXiv preprint 1705.04527*, 2017.

[3] Certicom. SEC 4 v1.0: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV). Technical report, Certicom Research, Canada, 2013.

[4] H. Cui, G. Karame, F. Klaedtke, and R. Bifulco. On the fingerprinting of software-defined networks. *IEEE Transactions on Information Forensics and Security*, 11(10):2160–2173, 2016.

[5] O. Flauzac, C. Gonzalez, A. Hachani, and F. Nolot. SDN based architecture for iot and improvement of the security. In *29th Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA)*, pages 688–693. IEEE, 2015.

[6] C. Gouvêa, L. Oliveira, and J. López. Efficient software implementation of public-key cryptography on sensor networks using the MSP430X microcontroller. *Journal of Cryptographic Engineering*, 2, 05 2012.

[7] S. Hayward, C. Kane, and S. Sezer. Operationcheckpoint: SDN application control. In *Int. Conf. on Network Protocols (ICNP)*, pages 618–623. IEEE, 2014.

[8] T. Kavitha and D. Sridharan. Security vulnerabilities in wireless sensor networks: A survey. *Journal of information Assurance and Security*, 5(1):31–44, 2010.

[9] S. Lim, S. Yang, Y. Kim, S. Yang, and H. Kim. Controller scheduling for continued SDN operation under DDoS attacks. *Electronics Letters*, 51(16):1259–1261, 2015.

[10] T. Luo, H. Tan, and T. Quek. Sensor OpenFlow: Enabling software-defined wireless sensor networks. *IEEE Comm. letters*, 16(11):1896–1899, 2012.

[11] A. Mahmud and R. Rahmani. Exploitation of OpenFlow in wireless sensor networks. In *Int Conf. on Computer Science and Network Technology (ICCSNT)*, volume 1, pages 594–600. IEEE, 2011.

[12] C. Margi, R. Alves, G. Nunez, and D. Oliveira. Software-defined wireless sensor networks approach: Southbound protocol and its performance evaluation. *Open Journal of Internet Of Things*, 4(1):99–108, 2018.

[13] H. Nguyen and M. Yoo. Analysis of link discovery service attacks in SDN controller. In *Int. Conf. on Information Networking*, pages 259–261. IEEE, 2017.

[14] K. Pelechrinis, M. Iliofotou, and S. Krishnamurthy. Denial of service attacks in wireless networks: The case of jammers. *IEEE Communications surveys & tutorials*, 13(2):245–257, 2011.

[15] M. Simplicio, M. Silva, R. Alves, and T. Shibata. Lightweight and escrow-less authenticated key agreement for the Internet of Things. *Computer Communications*, 98:43–51, 2017.

[16] C. Toledo, M. Simplicio, and C. Margi. A framework for building secure software-defined wireless sensor networks. *ENCOM*, 2019.