Imperial College London
Faculty of Engineering
Department of Computing

MONITORING THE HEALTH AND INTEGRITY OF WIRELESS SENSOR NETWORKS

RODRIGO VIEIRA STEINER

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London

July 2019

DECLARATION OF ORIGINALITY

I herewith certify that all material in this dissertation which is not my own work has been properly acknowledged.

Rodrigo Vieira Steiner

COPYRIGHT

ABSTRACT

Wireless Sensor Networks (WSNs) will play a major role in the Internet of Things collecting the data that will support decision-making and enable the automation of many applications. Nevertheless, the introduction of these devices into our daily life raises serious concerns about their integrity. Therefore, at any given point, one must be able to tell whether or not a node has been compromised. Moreover, it is crucial to understand how the compromise of a particular node or set of nodes may affect the network operation.

In this thesis, we present a framework to monitor the health and integrity of WSNs that allows us to detect compromised devices and comprehend how they might impact a network's performance. We start by investigating the use of *attestation* to identify malicious nodes and advance the state of the art by exploring limitations of existing mechanisms. Firstly, we tackle effectiveness and scalability by combining attestation with *measurements inspection* and show that the right combination of both schemes can achieve high accuracy whilst significantly reducing power consumption. Secondly, we propose a novel stochastic *software-based* attestation approach that relaxes a fundamental and yet overlooked assumption made in the literature significantly reducing time and energy consumption while improving the detection rate of honest devices.

Lastly, we propose a mathematical model to represent the health of a WSN according to its abilities to perform its functions. Our model combines the knowledge regarding compromised nodes with additional information that quantifies the importance of each node. In this context, we propose a new *centrality measure* and analyse how well existing metrics can rank the importance each sensor node has on the network connectivity. We demonstrate that while no measure is invariably better, our proposed metric outperforms the others in the vast majority of cases.

ACKNOWLEDGMENTS

I am deeply grateful to my supervisor Emil Lupu for all his dedication, guidance, and support. He has granted me numerous opportunities to grow as a researcher, was a constant source of wisdom and critical analysis, and has given me just the right amount of freedom being always present when necessary. I would also like to thank my examiners, Tim Watson and Christopher Hankin, for their time, constructive criticism, and the fruitful discussions we had during the Viva.

I am thankful to my fellow members of the Resilient Information Systems Security (RISS) Group for the many scientifical and technical discussions, for providing a pleasant working environment, and all the leisure moments we had together. A special thanks to Vittorio Illiano and Martín Barrère who had a more direct contribution to the work presented in this thesis, and also to Erisa Karafili for her friendship and support.

I am forever grateful to my family for their unconditional love and constant support. I would like to give a very special thanks to my parents, my sister, and my grandparents. Last but not least, my wife Andréa Puerta Pereira Oliveira has all my gratitude for being always by my side, a constant source of encouragement, with all her joy, enthusiasm, love, and affection.

This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under Grant No. 206707/2014-2.

To my family.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Wireless Sensor Networks (WSNs) are distributed systems composed by dedicated computer devices, named sensor nodes, that are spatially scattered across an environment to monitor physical phenomena like humidity, luminosity, temperature, among others. Such sensors nodes work autonomously and collaboratively to relay their data towards a base station, also known as sink node, which works as a gateway connecting the network to the outside world. These networks have a wide range of applications [1], and, over the last years, have been perceived as a fundamental technology that will pave the way to the Internet of Things (IoT) [2].

While the adoption of WSNs and development of the IoT can undoubtedly bring numerous benefits, the ingress of the so called *smart* or *connected* devices across homes, industries, healthcare systems, and critical national infrastructure raises serious concerns regarding the security of such systems. Compromised devices can plunder confidential information, disseminate spurious data, or even interrupt services, all of which could have tragic repercussions. Nonetheless, it is highly likely that such devices are going to be attacked and compromised at some point during their lifetime. We have already seen cases such as Stuxnet, a malware targeting industrial control systems [3], the Mirai botnet, mainly constituted of IoT devices and believed to be the biggest distributed denial of service (DDoS) attack to date [4], and more recently the WannaCry ransomware hitting organizations such as the British National Health Service (NHS) [5]. Therefore, at any given point, one must be able to tell whether or not a device is operating as it should or if it has been compromised. Moreover, it is crucial to understand the impact the compromise of a particular node or set of nodes can have on the network performance and the system as a whole.

Most of the related work on the security of WSNs focuses on securing its underlying protocols [6] or on the trust and reputation management [7] of nodes in the network. However, by exploiting existing software vulnerabilities, an adversary can easily compromise high reputation nodes without breaking their protocols. As observed by Gu and Noorani [8], vulnerabilities arising from low-level memory faults, such as stack overflows [9], and more recent, sophisticated, exploitation techniques, such as return-oriented programming (ROP) [10, 11], pose a real threat to WSNs. In fact, these vulnerabilities have already been exploited in the form of code injection attacks [12] and self-propagating mal-packets [8]. Thus, a mechanism to verify the integrity of sensor nodes is necessary. The integrity of a node is a binary property that indicates whether it has been modified in an unauthorized manner or not [13]. Compromised nodes cannot be trusted as they may malfunction or present malicious behavior.

WSNs can comprise hundreds or thousands of sensor nodes that are distributed in the environment. In many cases, nodes are deployed in an unsystematic fashion making it difficult to know their exact location. While it is not practical for most application scenarios, for some of them it is not even possible to physically reach and verify the integrity of each node composing the network. Thus, there is a need for a mechanism that not only verifies the integrity of the nodes but a mechanism that can do this verification remotely, without physical access to the device being verified.

Many challenges arise under these circumstances: how to scale the verification for the high number of devices; the heterogeneous hardware and software architecture of such devices; the limited amount of energy they have available; and the intrinsically unreliable wireless medium in which they communicate. Nevertheless, several attestation mechanisms have been proposed over the last years to identify compromised devices by verifying their software integrity [14]. However, existing approaches make different assumptions over the system and adversary models. Consequently, it is difficult to compare them and analyse the security properties provided by these schemes. Moreover, this lack of coherence has already resulted in new proposals [15, 16, 17, 18, 19, 20, 21, 22, 23] being vulnerable to formerly known attacks [24].

Unlike the integrity of a single device, the *health of a network* is not a binary property indicating whether it has been compromised or not, but rather an indicator of how well it can operate in its current state and fulfil its functions. To answer that, one must be able to tell not only which nodes have been compromised, but also how important each node is to the network's operation. While the importance of a node is a reflection of the tasks it executes, the significance of each task is application specific and varies from case to case. The role a node plays on the network connectivity, however, depends primarily on its topological position and can be generalized.

A number of centrality measures have been proposed in the network theory literature to identify the most important nodes within a network [25]. However, each measure has its own interpretation of what makes a node important. Consequently, they may provide distinct outcomes, and there is no optimal measure that best suits all scenarios.

1.1 GOAL AND SCOPE

The main goal of this thesis is to develop a framework to monitor the health and integrity of WSNs that allows us to identify compromised nodes and to understand how well the network can operate even in the presence of compromise. To achieve this goal we first investigate and propose two improvements to advance the state of the art of attestation techniques towards more practical solutions. We then examine how well centrality measures rank the impact sensor nodes have on the network connectivity, and elaborate a mathematical model capable of combining all this information and expressing the health of a network as a single value indicating its operational level.

There are many ways an adversary can compromise nodes in a WSN. While we assume an attacker that modifies the software running on the network nodes, the exact steps necessary to perform such attack are out of scope of this thesis. Similarly, many actions can be taken once a malicious device is detected. While our health model helps us to understand the impact a compromise has on the network and may serve as a guide to take more effective responses we leave this as future work.

1.2 SUMMARY OF CONTRIBUTIONS

The main contributions of this thesis are:

Attestation literature survey. Over the last years, several attestation techniques have been proposed. While they all use variants of a challenge-response protocol, they make different assumptions about what an attacker can and cannot do. Thus, they propose intrinsically divergent validation approaches. We survey the different approaches to attestation focussing in particular on those aimed at WSNs. We discuss the motivations, challenges, assumptions, and attacks of each approach. We then organise them in a taxonomy and discuss the state of the art, carefully analysing the advantages and disadvantages of each proposal. We also identify open research problems and give directions on how to address them.

Combining attestation with measurements inspection. Attestation and measurements inspection are different security techniques but can be used complementary towards the same goal: ascertaining the integrity of sensor nodes in WSNs. We compare the benefits and drawbacks of both techniques and seek to determine how to best combine them. However, our study shows that no single solution exists, as each choice introduces changes in the measurements collection process, affects the attestation protocol, and gives a different balance between the high detection rate of attestation and the low power overhead of measurements inspection. Therefore, we propose three strategies that combine measurements inspection and attestation in different ways, and a way to choose between them based on the requirements of different applications. We analyse their performance both analytically and in a simulator. The results show that the combined strategies achieve a detection rate close to attestation, in the range 96-99%, whilst keeping a power overhead close to measurements inspection, in the range 1-10%.

Stochastic software-based attestation. Existing software-based attestation proposals rely on strong assumptions that hinder their deployment and might even weaken their security. One such assumption is that using the maximum known network round-trip time to define the attestation timeout allows all honest devices to reply in time. While this is normally true in controlled environments, it is generally false in real deployments and especially so in a scenario like WSNs where numerous devices communicate over an intrinsically unreliable wireless medium. Moreover, a larger timeout demands more computations, consuming extra time and energy and restraining the untrusted device from performing its main tasks. We review this fundamental and yet overlooked assumption and propose a novel stochastic approach that significantly improves the overall attestation performance. Our experimental evaluation with devices communicating over real-world networks demonstrates the practicality and superior performance of our approach. When compared with the current state of the art solution, we reduce the total attestation time and energy consumption around seven times for honest devices and two times for malicious ones, while improving the detection rate of honest devices (8% higher true positive rate) without compromising security (0% false positive rate).

Network health model. We propose a model to represent the health of WSNs that allows us to evaluate a network's ability to execute its functions even in the presence of malicious devices. Our model combines the knowledge regarding which nodes have been compromised with additional information that quantifies the importance of each node. In particular, we investigate how well different centrality measures identify the significance of each node for the network connectivity. In this process, we propose a new metric named *current-flow sink betweenness*. Through a number of experiments, we demonstrate that while no measure is invariably better in identifying sensors' connectivity relevance, the proposed current-flow sink betweenness outperforms existing metrics in the vast majority of cases.

1.3 PUBLICATIONS

This dissertation is based on the following publications:

Rodrigo Vieira Steiner and Emil Lupu. Attestation in Wireless Sensor Networks: A Survey. *ACM Computing Surveys*, 2016

Vittorio Paolo Illiano, Rodrigo Vieira Steiner, and Emil Lupu. Unity is strength! Combining Attestation and Measurements Inspection to handle Malicious Data Injections in WSNs. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (WiSec '17), 2017

Rodrigo Vieira Steiner, Martín Barrère and Emil Lupu. WSNs Under Attack! How Bad Is It? Evaluating Connectivity Impact Using Centrality Measures. In *Living in the Internet of Things: Cybersecurity of the IoT*, 2018

Rodrigo Vieira Steiner and Emil Lupu. Towards more practical software-based attestation. *Computer Networks*, 2019

1.4 THESIS OUTLINE

The remainder of the thesis is structured as follows. In Chapter 2, we provide a comprehensive review of attestation mechanisms in the context of WSNs and introduce fundamental concepts for the remainder of the thesis. Then, in Chapter 3, we show how attestation can be combined with measurements inspection to obtain a high detection rate with low overhead. Next, in Chapter 4, we introduce a novel stochastic software-based attestation approach that targets a fundamental assumption made by existing mechanisms significantly improving the overall attestation performance. In Chapter 5, we present a model to express the health of WSNs, propose a new centrality measure and analyse how well different measures identify the significance of each node for the network connectivity. Finally, we conclude this thesis in Chapter 6 summarizing our achievements and discussing future work.

2 ATTESTATION

Over the last years, several attestation techniques have been proposed in the literature to detect malicious devices by validating their software integrity. In this chapter, we extensively analyse the state of the art of such mechanism in the context of WSNs. We start in Section 2.1 by giving an overview of attestation, followed by a comprehensive analysis of the system and adversary models. We then cover common attacks and assumptions made, and their relevance in the context of WSNs. In Section 2.2 we introduce a taxonomy that captures the fundamental differences between existing solutions. Each distinguishing characteristic used in our taxonomy is illustrated with a representative example taken from the state of the art, which allow us to evaluate the advantages and disadvantages of existing approaches. Following this analysis, we discuss open research problems and give directions on how to tackle them in Section 2.3. Finally, we conclude the chapter in Section 2.4.

2.1 OVERVIEW

A typical attestation mechanism follows a challenge-response protocol, as shown in Figure 1. A trusted device verifies the integrity of an untrusted device which has to prove its innocence. The devices are commonly named after their roles: *Verifier* and *Prover*, respectively. The goal of the attestation procedure is to allow an honest, non-compromised, *Prover* to generate a response that assures the *Verifier* that the prover is in a legitimate state [26]. A compromised *Prover* will either generate invalid responses or will not be able to generate a valid response within an expected time limit.

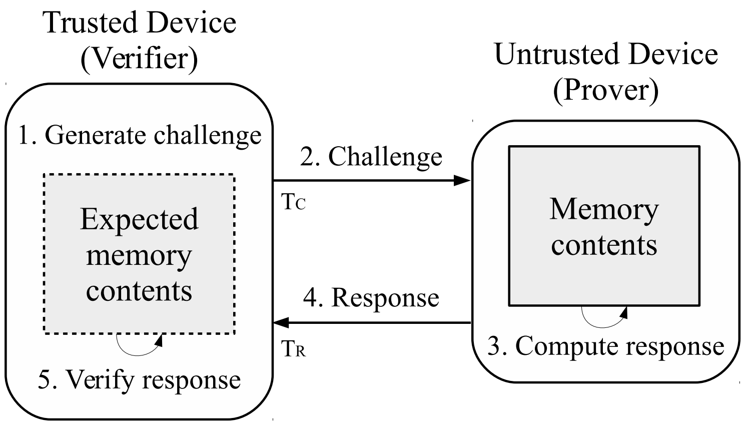


Figure 1: Attestation overview.

It is generally assumed that the *Verifier* knows, in advance, the correct internal state — the memory contents — of the *Prover*. Therefore, the *Verifier* challenges the *Prover* to demonstrate that it is in a valid, expected, state. The *Prover* then executes an attestation routine, which will compute and send back a response based on the challenge received from the verifier and its internal state. The *Verifier* compares the answer received from the prover with the expected one, and if there is a match then it can assert that the device has not been compromised. However, the *Verifier* only awaits the prover's response for a limited amount of time T_A, which must be at least as long as the time taken by the *Prover* to execute the genuine attestation routine. If the time difference between receiving the response and issuing the challenge, $T_R - T_C$, exceeds T_A, the *Verifier* knows something may be wrong with the *Prover*. Note here, that when performing attestation over a network, the time delay to send and receive messages must also be taken into account. As we will discuss further in Section 2.2, there are two different approaches regarding timing control: *strict* and *loose*.

We are not the first to analyse the attestation process. Nonetheless, existing analyses are built under different assumptions. For example, Datta et al. [27], Coker et al. [28] and Francillon et al. [26] assume that the verifier and the prover need to share some secret information, e.g., a cryptographic key. They presume that this secret is defined prior to network deployment and that there exists some hardware support preventing the adversary from accessing it. Their models, however, also do not consider timing requirements. On the other hand, Armknecht et al. [29] and Li et al. [30] focus on software-based attestation techniques that assume no hardware support and are only concerned with attesting the program's memory. The analysis by Li et al. [30] also covers approaches that completely erase the prover's data memory. However, neither of them cover proposals that attempt to verify the data memory instead of simply wiping out its contents.

In this work we follow in the footsteps of existing analyses, however, we aim to provide a more comprehensive model capable of encompassing all approaches relevant for WSNs, which allows us to evaluate the tradeoffs between different techniques. In this context, the intrinsic characteristics of WSNs play an important role. The simplified hardware and software architecture of sensor nodes has both a positive and negative side. As a consequence of their reduced storage space, there is less memory available for an adversary to explore. Furthermore, differently from general-purpose computers, each node has a well-defined application to execute. Thus, it is easier to know what are the expected memory contents of these devices. Moreover, sensor nodes are usually equipped with single-core processors and have a single flow of execution, which reduces the possibilities of an adversary to perform parallel operations during attestation. On the downside, the limited amount of energy powering sensor nodes is certainly one of the biggest restrictions imposed. The fact that the sensors may be placed in hostile environments and communicate over a wireless channel which may suffer from external interference is another complication.

2.1.1 *System Model*

There are three entities that need to be considered when modelling an attestation mechanism: the verifier V, the prover P, and the adversary A. Although attestation mechanisms can be applied to other scenarios, we focus here only on WSNs. Therefore, both the verifier and the prover are wireless sensor nodes. The verifier can possess more computational power than common network nodes, but this is not mandatory. Meanwhile, the adversary

Table 1: Notation summary.

Term	Description
A	Adversary
c	Challenge
M	Memory
M_e	External memory
M_p	Program memory
M_d	Data memory
M_{mmio}	Memory Mapped Input/Output
M_r	Registers
P	Prover
r	Response
S	Internal state
T_A	Attestation time limit
T_C	Challenge sending time
T_R	Response reception time
V	Verifier

is either launching attacks remotely or using an already compromised network node. Table 1 summarizes the notation adopted in this chapter.

A prover P has an internal state $State(P) = S$ that reflects the contents of its memory M. Ideally, all memory contents should be attested including the program memory M_p, data memory M_d, registers M_r, MMIO M_{mmio}, and even external memories M_e. However, each of the different attestation mechanisms covers different sections of the memory, and in practice, some parts of M are left unverified. For instance, Spinellis [31] only checks M_p, while Zhang and Liu [32] partially validate M_d and nothing else. For simplicity, we consider (as reflected in the adopted notation) that the state S of a prover P corresponds only to the portions of memory being attested.

One might think that the attestation is safer when a larger amount of memory is being covered. However, this is not necessarily so. Usually, M_p is far greater than M_d but some attacks only need to alter M_d to succeed [10, 11]. Therefore, a safer approach would be to attest all different types of memory [24]. However, even when all memories are covered, an adversary can still perform several types of attacks. In practice, attestation only provides a probabilistic guarantee of the integrity of a prover. In this work, we aim to identify the factors that influence this probability.

An attestation process has three main constituents: CHALLENGE, ATTEST, and VERIFY, which are present in all the approaches described in the literature although their implementation may vary. Below we discuss each of them individually and highlight the design principles that must be followed to provide secure attestation. As stated before, we adopt much of previous analyses [27, 28, 29, 30, 26] while centering the discussion around WSNs.

CHALLENGE() This procedure is executed by the verifier. It outputs a *random* challenge c that is transmitted to the prover and which may contain a nonce, a timestamp, memory addresses, or even the attestation routine to be executed by P. CHALLENGE must follow three design principles: *Authenticity*, *Freshness*, and *Unpredictability*. The first two principles are essential to prevent an adversary from performing Denial-of-Service (DoS) attacks by unrestrictively forcing P to perform ATTEST. Nevertheless, DoS attacks are not a major concern in attestation of sensor nodes as there are easier methods to achieve the same effect such as channel jamming. The last principle prevents the adversary from calculating the result of the attestation routine in advance since ATTEST either takes as input or is itself, the unpredictable challenge generated. Formally, this could be described as: there exists no efficient algorithm Alg such that, for non-negligible ϵ,

$$Pr[Alg()_i = Challenge()_{i+1} : i \in \mathbb{N}^+] > \epsilon.$$

ATTEST(S, c) The attestation routine is executed by the prover. It takes as input the state S of P and the challenge c sent by V. When c is the attestation routine itself, it is downloaded and installed in a pre-defined memory space in M_p. The goal of ATTEST is to compute a small attestation response r, which must directly be based on both S and c. This drastically reduces the amount of data transmitted from P to V as otherwise the prover would have to transmit the entirety of its memory contents to the verifier to prove its integrity. ATTEST must adhere to five design principles. *Authenticity* allows the verifier to confirm the source of r. *Atomicity* guarantees that ATTEST is not interrupted during execution preventing the adversary from modifying S, moving the malware around to avoid detection, or parallelizing the computation. *Unforgeability* prevents an adversary from producing the same response r, at least not faster than ATTEST. *Dynamicity* in the sense that r should reflect the actual running system, and not just some static part of the memory. Finally, *Determinism* enables the verifier to reach the same result r on its own. Formally:

$$Pr[\exists\ S \neq S' : Attest(S,c) = Attest(S',c)] \leq \epsilon$$

VERIFY($E[S]$, c, r, T_A, T_C, T_R) This function, executed by the verifier, takes as inputs the expected state of the prover $E[S]$, the challenge c that the verifier has generated, the response r from P, the attestation time limit T_A, the time when the challenge has been sent T_C and the response received T_R. VERIFY must respect one design principle: *Determinism* so that it *accepts* iff r reflects both S and c and $T_R - T_C \leq T_A$. Furthermore, VERIFY must always *accept* responses from uncompromised provers. Formally:

$$\begin{aligned} Pr[Verify(E[S],c,r,T_A,T_C,T_R) = accept | State(P) = S \\ \wedge\, r = Attest(S,c) \\ \wedge\, T_R - T_C \leq T_A] = 1 \end{aligned}$$

Having identified the main components, we can now formalize the interactions between the verifier and the prover. Figure 2 shows all steps described so far, which are present in all existing approaches. In some approaches, however, these interactions may slightly differ. For example, they can be preceded by an additional step where the prover itself requests to be verified. Park and Shin [33] demand that each sensor proves its integrity to a verification server before accessing the network, so new sensors must ask to be attested before using network resources. AbuHmed et al. [34] provide another example of variation of the interactions: to verify the freshness and authenticity of a challenge request, a prover

$$\begin{aligned}
V &: c \overset{R}{\leftarrow} Challenge() \\
V \rightarrow P &: c \\
V &: T_C \leftarrow current\ time \\
P &: r \leftarrow Attest(S, c) \\
P \rightarrow V &: r \\
V &: T_R \leftarrow current\ time \\
V &: Verify(S, c, r, T_A, T_C, T_R)
\end{aligned}$$

Figure 2: Generic attestation procedure.

sends the challenge, encrypted together with a random number, back to the verifier. The verifier then decrypts the message and sends back to the prover the encrypted random number.

2.1.2 *Adversary Model*

The objective of an adversary is to compromise a sensor node without being detected by the attestation procedure. We do not address techniques used to compromise nodes — the interested reader may refer to [8, 12, 35] for additional information on this topic — but rather cover the attacks an adversary may perform to overcome attestation. As observed by Armknecht et al. [29], the adversary has two different phases to perform an attack: an initial phase that occurs before the attestation begins, and a second one that starts when the prover is challenged. Before attestation, the adversary can use unlimited resources and may modify the state S of a prover at will, resulting in a new state S'. However, after being challenged the adversary has a limited amount of time to send a response.

2.1.2.1 *Attacks*

A series of attacks are typically portrayed in the literature on how an adversary may subdue the attestation process after having compromised a device. We describe each of them individually below.

Precomputation. An adversary can precompute all operations of the attestation routine that are not influenced by the challenge, thus gaining time that may be used to execute other operations during the attestation. Furthermore, if the adversary can predict the challenges generated by the verifier, it can then precompute valid responses.

Replay. An adversary can eavesdrop a valid attestation response from a non-compromised node, store it, and retransmit the message when challenged by the verifier. Since the response reflects both the prover's internal state and the challenge, this approach can only work if both nodes execute the same program and receive the same challenge.

Forgery. The adversary can attempt to generate a valid response despite modifications in the prover's internal state. This can be achieved by executing a modified version of the attestation routine or by altering memory contents in such a way that modifications neutralize one another during response computation [24].

Memory copy. If there is enough free space in memory, an adversary can store its malicious data and still keep a copy of the original memory contents [36]. Then, the adversary can modify the attestation routine so that it computes a response over the memory locations where the original contents copy is maintained.

Data substitution. This attack is a special case of the memory copy attack and occurs when there is not enough space to keep a full copy of the original memory contents. The adversary can then modify part of the original contents and keep a copy only of the overwritten data [37]. In this case, the attestation routine must be adjusted to redirect memory reads from altered memory addresses to the original contents' copy location.

Compression. In another special case of the memory copy attack an adversary can compress the original contents to obtain enough space to store its malicious data [24]. During attestation, the adversary can then decompress the original contents on-the-fly to compute the expected response.

High execution time variance. The execution time of the attestation routine may vary due to clock, cache, and translation lookaside buffer (TLB) fluctuations [38]. The higher the number of computations performed, the higher the time variance. To avoid the case where honest provers are not able to reply in time, the challenge timeout should also include the execution time jitter. Therefore, an adversary can execute additional instructions and still reply within the timeout with non-negligible probability.

Collusion. Compromised nodes can act together to compute valid responses. For example, multiple nodes that execute the same application can install the malware in different memory locations [39]. Then, during attestation, the nodes can exchange messages to recover the original memory contents. Another possibility is to divide the attestation routine operations across multiple devices to speed up the computation of the response.

Impersonation. An adversary can take multiple identities and impersonate other nodes (also known as Sybil attack [40]). In doing so, it can masquerade as a genuine node during attestation and send an invalid response, thus making the verifier believe the original node has been compromised. Besides, a compromised node may also impersonate the verifier and forward the challenges it receives to a genuine node and then forward the correct response back to the verifier.

Proxy. This attack is a special instance of collusion and impersonation attacks because it requires a device with better computing capabilities than the prover. Whenever a compromised node is challenged, it forwards the challenge to this proxy device. The proxy keeps a copy of the node's original memory contents and is able to impersonate it to compute a valid response. Since the proxy device has greater computing resources, it can also compute the attestation routine faster than common nodes.

Code reuse. Using techniques such as Return-oriented Programming (ROP) [10, 11], an attacker can use existing code, without altering it, to execute malicious operations. By linking together small sequences of instructions, called *gadgets*, present in existing programs, it is possible to perform arbitrarily complex operations. Originally, each gadget would end with a *ret* instruction, and the attacker could chain different gadgets together by modifying the stack, making them execute one after another. It was shown subsequently that these attacks can be performed without using *return* operations, but also through other instructions that alter the program control flow, such as *branch*, *call*, and *jump* [41]. Because gadgets are built from original program instructions, ROP attacks can circumvent defenses that assume the adversary must modify or insert new code [11].

It is interesting to see that the majority of attacks do not require the malicious node to interact with other devices. Only the *Replay*, *Collusion*, *Impersonation*, and *Proxy* attacks require such interaction.

2.1.3 *Assumptions*

We present here the most common assumptions encountered in the literature and examine existing exceptions and conflicts among them.

The verifier cannot be compromised by the attacker. In many cases, the verifier is the network's base station. Since the base station acts as a gateway connecting the sensor network to the outside world, it is common to assume that it cannot be compromised [42]. Nonetheless, not all approaches make this assumption. For instance, Yang et al. [39] proposes a distributed attestation scheme where all nodes in the network can play the role of the verifier. However, since all nodes are vulnerable to compromise, the attestation process no longer rely on a single verifier. Instead, multiple neighbors of a node collaborate to attest it. In this case, the attestation result also depends on how many devices an attacker has compromised.

The verifier knows the expected state of the prover. To attest that a device has not been compromised a verifier must know what to expect from such device. In most approaches, the verifier has complete knowledge of the software that should be running on the prover device [43]. With this knowledge, the verifier can know the set of valid states a prover can be. A different strategy is to issue each node with a certificate of its valid configuration [20]. This allows the verifier to attest a device without knowing its settings in detail.

The verifier knows the hardware architecture of the prover. The prover's hardware plays a major role in the construction of the attestation routine ATTEST. It defines which operations must be performed by the prover to demonstrate its integrity. For instance, ATTEST can take advantage of any tamper-resistant hardware available to perform its operations and protect secret information. On the other hand, when no such hardware is available the attestation routine has to be carefully designed. In such scenarios, ATTEST is usually assumed to be optimal so that an adversary cannot optimize it and execute further operations to hide its modifications and still reply in a valid time. Therefore, the verifier must know the microcontroller, clock speed, Instruction Set Architecture (ISA), and memory architecture of the prover [44].

The adversary can reverse engineer the prover's software and hardware. Commodity sensor nodes [45, 46, 47] do not provide any tamper-resistant hardware since such nodes are supposed to be cheap and small in size. So the node's software is usually stored in unprotected memory, which an attacker can read, reverse engineer, and modify. Therefore, the attestation procedure cannot rely on secret information such as cryptographic keys that would be stored in unprotected memory. Needless to say, this assumption does not hold for mechanisms developed targeting devices with tamper-resistant hardware.

The adversary has full control of the prover's memory. In the absence of hardware controls to protect the prover's program memory, an adversary can modify the underlying software at will. Consequently, the attacker can control the prover and all its communications and can perform both passive and active attacks, such as eavesdropping, packet injection, replay, selective forwarding, and many others. Some mechanisms explore the use of tamper-resistant hardware, Read Only Memory (ROM), or even a Memory Protection Unit (MPU) to limit the adversary control.

The adversary cannot modify the prover's hardware. It is generally assumed that an attacker cannot perform any hardware modification to the sensor node, such as attaching more memory, altering memory access timing, or even changing the processor clock speed [44]. This assumption is usually justified in terms of the cost and practicality of the attack. Not only would an attacker require physical access to the sensor to modify its hardware but also the means and time to carry out the modification on a significant subset of the sensors in the WSN. To remain consistent with this assumption one must also consider that in such cases the attestation mechanism must similarly not rely on physical access to validate the integrity of the node and must be done remotely. Note that attestation works as a first line of defense, compelling the adversary to either modify individual sensors or deploy new, already modified, sensors in the network. As observed by Park and Shin [33], other techniques like intrusion detection systems [48, 49, 50] can be deployed in conjunction with attestation to defend against such attacks.

2.2 TAXONOMY

Over the last years, several attestation mechanisms have been proposed. Nevertheless, as seen in Section 2.1, they make different, and sometimes conflicting, assumptions. Therefore, they use intrinsically different ways of achieving their goals. The current literature most commonly separates attestation into two categories: *hardware-* and *software-based*. However, we do not believe this classification is always helpful as it hinders the comparison of existing approaches in terms of the security properties achieved. There is more to the process of attestation than the use or not of tamper-resistant hardware. In this section, we review the main characteristics of existing approaches and propose a new taxonomy to classify the different techniques proposed. This allows us to compare their advantages, disadvantages, and vulnerabilities. Our proposed taxonomy, shown in Figure 3, identifies eight major characteristics of attestation mechanisms, each of which can be realized in different ways. We discuss them in more detail below.

2.2.1 *Evidence Acquisition*

Arguably, the biggest issue in attestation is the manner in which the verifier draws evidence of the prover's integrity and the extent to which this evidence can be trusted. Three main different approaches can be identified in the literature: *hardware-based*, *software-based*, and *hybrid* techniques.

Hardware-based techniques rely on tamper-resistant hardware such as the Trusted Platform Module (TPM) [51] or Physical Unclonable Functions (PUFs) [52, 53]. For example, Tan et al. [18] describes a TPM-enabled Remote Attestation Protocol (TRAP) for WSNs in which all sensor nodes are equipped with TPMs responsible for securing preloaded secrets. Prior to the network deployment, each node is preloaded with cryptographic keys to safely communicate with neighboring nodes and the base station. When a node is powered on, it

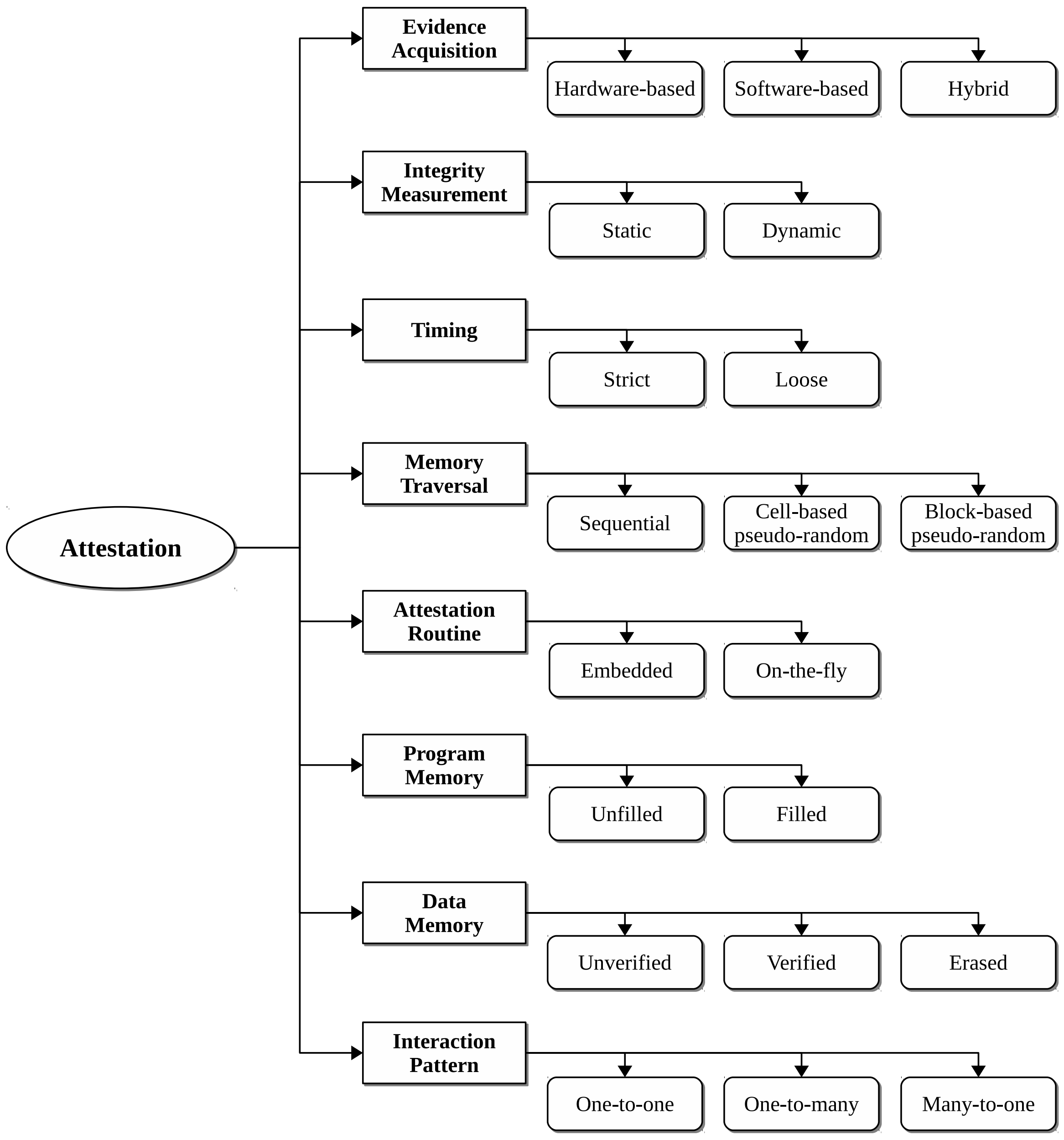


Figure 3: A taxonomy of attestation mechanisms.

transfers control to its bootloader, which, differently from the application, cannot have its code updated after node deployment. Therefore, the bootloader code is used as a first line of defense. During the initialization phase, each node computes a hash of the bootloader code, stores it into a TPM Platform Configuration Register (PCR) and uses it to seal the cryptographic keys into the TPM. When the bootloader is running, it computes and stores a hash of its own code. Consequently, if the bootloader code has been altered, the TPM unseal command will fail, and the node will be unable to generate a valid attestation response. The bootloader also computes and stores a hash of the application code into the TPM. During attestation, the verifier, which can be any neighbor of the prover, uses the TPM to generate a random number and encrypts it using the key shared with the prover. It then sends the challenge to the prover and asks the base station for the value of the prover's hash code and its public key. Upon receiving the challenge, the prover has to decrypt it, pass it as a nonce to the TPM, and construct a response based on the TPM output. This response can only be correct if the application code has not been altered. However, this approach only verifies the prover's M_p. Therefore, it is vulnerable to ROP attacks, which only need to alter the call stack to succeed.

Software-based techniques do not rely on secure hardware. Instead, the prover executes an attestation routine that produces an allegedly unforgeable result. The work by Spinellis [31] is one of the earliest on *software-based* attestation. The author proposes the use of reflection to perform software integrity verification, and although the proposed approach is not specifically designed for WSNs, its computational performance remains within the practical limits of WSN nodes. In this approach, the verifier randomly chooses two overlapping memory regions of the prover's M_p, such that one region covers the initial memory addresses and the other covers the last addresses, and they overlap somewhere in-between. The prover then computes a cryptographic hash for each region. The verifier similarly calculates the corresponding hash values, from its own copy of the memory being attested, and compares them with the values received. Because the hash values cover the entire M_p any modification to it will be detected. This, however, is based on the assumption that the attacker cannot interrupt the verification process and move the malware around, always relocating it to somewhere out of the current hash computation range. Moreover, as the hash computations are independent of one another, colliding nodes could compute them separately and in parallel to avoid detection by timing differences. Spinellis also describes an extension of this procedure where the prover also sends data regarding its processor state, such as the contents of the CPU cache or a hardware performance counter. However, as this information is not used in the hash computation, once an adversary eavesdrops a valid response, it could simply extract this part and replay it.

We classify as *hybrid* techniques, approaches that do not depend on a tamper-resistant hardware but do require specific hardware, such as ROM, to achieve attestation. For example, Perito and Tsudik [54] present a secure code update mechanism for embedded devices based on *Proofs of Secure Erasure (PoSE)*. This procedure requires a small amount of ROM to store the attestation routine and thus prevents an adversary from modifying it. The verifier sends incompressible random noise large enough to completely fill the prover's writable memories. The prover, uses the code stored in ROM, to compute a Message Authentication Code (MAC) over all data received, using the last bits as the key, and sends it back to the verifier, which checks it. This procedure is also particularly targeted at secure code updates as the "incompressible random noise" can be an encrypted form of the new code for the node. Once the verifier has verified the MAC, it can then send to the prover the key used for encryption, which the prover uses to decrypt the code and perform the code update. The main disadvantage of this approach is the high communication overhead it introduces. The need to transmit enough data to completely fill the prover's writable memories consumes significant amounts of time and energy. Furthermore, there is an implicit assumption that nodes under attestation cannot collude.

2.2.1.1 *Discussion*

Tamper-resistant hardware works as a root of trust, and all the information and services it provides are considered to be reliable thus facilitating the attestation procedure. However, a major drawback of *hardware-based* techniques is that they cannot be used on legacy devices that do not have such hardware. Furthermore, using tamper-resistant hardware increases both sensor cost and energy consumption making it inappropriate in a number of scenarios. The assumption that tamper-resistant hardware is sufficient to engender trust is also increasingly threatened as numerous types of attacks can still be explored by an adversary [55]. Side-channel attacks such as timing [56], power [57], and electromagnetic analysis [58] are some examples. More recently, both TPMs [59, 60, 61, 62, 63] and PUFs [64, 65, 66] have been the target of attacks. In contrast, *software-based* techniques do not rely on tamper-resistant hardware. The benefits of these approaches are that they can be applied to legacy devices and do not increase the node's cost and size. Therefore, it is unsurprising that the majority of attestation mechanisms proposed for WSNs are *software-based*. *Hybrid* approaches share advantages and disadvantages of *hardware-* and *software-based* techniques. For example, writing the attestation routine in ROM guarantees that it will not be modified by an attacker. However, it is dependent on having a sufficient amount of space available in the ROM, and this may not be the case with legacy devices. Furthermore, since physical attacks are difficult and costly, the use of tamper-resistant hardware might be considered overkill. For instance, a MPU can be used instead, to prevent illegitimate accesses to secrets [67, 68].

There is much discussion on the feasibility of remote attestation using solely *software-based* techniques. Before these schemes started being applied to WSNs — which theoretically facilitates their implementation, as sensor nodes have a much simpler architecture — Kennell and Jamieson [43] proposed to use CPU execution side effects, such as TLB misses, into a genuine test. The viability and reliability of using such side effects were then significantly debated [69, 70, 71]. Castelluccia et al. [24] investigates the shortcomings of existing approaches to embedded devices and presents two generic attacks, which have since been refuted [72] and then reasserted [73]. They argue that it is very difficult to correctly design attestation schemes with *strict timing* conditions because their implementation must be highly optimized. They also claim that, contrary to some existing schemes, all memories of the prover must be attested and that doing so is quite challenging. Francillon et al. [26] asserts that *software-based* techniques can be secure only if the prover and verifier communicate directly with no intermediate nodes, and thus cannot be used to perform attestation *across a network*. However, Yang et al. [19] present a delay-resilient software-based attestation mechanism capable of performing multi-hop attestation named Low-cost Remote Memory Attestation (LRMA). LRMA demands the response packet to go through the same path taken by the challenge. Relay nodes record the time when they receive each packet and report it to the verifier which can then estimate the average single-hop delay and detect compromised nodes by using a Bayesian classifier. Moreover, if the network is using a Time Division Multiple Access (TDMA) based Medium Access Control (MAC) [74] then the network delay is known. Furthermore, it is also possible to attest all network nodes without using multi-hop attestation. For example, Seshadri et al. [42] propose an expanding ring method, on which the base station starts by attesting nodes one hop away from it and then asks these nodes to attest their neighbours. The verification then proceeds in a hop-by-hop manner resembling an expanding ring. Another alternative is presented by Yang et al. [39] where the authors propose a *many-to-one attestation* in which the prover's neighbors execute the verification procedure avoiding the need for multi-hop attestation.

2.2.2 *Integrity Measurement*

The internal state of a prover can comprehend its program memory, data memory, registers, MMIO, and even external memories. Memories can be further divided into a *static* part whose contents never change during normal software execution, and a *dynamic* part whose contents may be inserted, removed or modified.

Static integrity measurement approaches verify only the *static* part of a prover's memory. For example, in the lightweight attestation scheme for WSNs presented by Kiyomoto and Miyake [17] all nodes have their memory divided into two parts: a program area M_P for storing program code and data, and another area M_A for attestation. Both M_P and M_A are divided in what the authors named *registers* (but without discussing the size of a register — it thus could be a single memory address or a block of addresses). If the program code is not large enough to fill M_P further random data is used in order to fill it. During the initialization phase, a sensor node randomly selects a register from one of its neighbors, computes a hash value for it, and stores the result in M_A. The node then repeats this process, randomly selecting different registers from different neighbors until it fills M_A. During attestation, a node randomly chooses a register either from M_P or M_A. If it selects a register from M_P, then the node gets the corresponding M_A register stored in one of its neighbors. Otherwise, if it selects a register from M_A, then it gets the corresponding M_P register stored in one of its neighbors. In both cases, the hash is recalculated from the M_P register, and the result is compared with the M_A register. If the values do not match, one or both nodes have been compromised. However, there is no way to find out which node has been compromised and, therefore, both nodes must terminate their operation. A terminated node stops communicating with the network to avoid the propagation of malicious code. It is, however, possible for an adversary that successfully compromises a node to modify the code in such a way that it never terminates, even if it does not pass attestation. Furthermore, the attestation verifies only one register at a time, so it has poor memory coverage and, for example, an adversary that modifies only one register has a good chance of remaining undetected.

Dynamic integrity measurement approaches check run-time properties of the software executing on the prover where such properties represent the behavior that must be satisfied during the normal execution of a program. For instance, the stack frames are arranged as a linked list, where the base pointer of a frame points to the base pointer of the previous frame. Remote Dynamic Attestation System (ReDAS) [75] is an example of such an approach. It automatically extracts the properties from each application's source code and binary in an offline phase. Then, during the program execution, any integrity violation evidence is recorded. To prevent an adversary from modifying the recorded evidence, every prover is equipped with a TPM. Therefore, when a violation is detected it is immediately sealed into the TPM. Then, when challenged by a verifier, all the prover has to do is send the sealed information. However, ReDAS only checks a subset of all possible dynamic system properties and measures the system integrity only at system calls. Therefore, an adversary can still succeed if it only changes properties not covered by ReDAS, or if it hides modifications between system calls.

2.2.2.1 *Discussion*

To check if a device has been compromised, the verifier must know in advance the set of valid states for the device. Since contents in *static* memory regions do not change during normal software execution, they provide a straightforward way to attest a device. The verifier challenges the prover to calculate a checksum over these memory regions and come up with a valid response, which turns out to be difficult to achieve unless these memory regions have not been modified. Difficult, but not impossible. As we have seen, there are numerous methods an adversary may use to circumvent attestation, e.g., forgery, memory copy, and collusion attacks. If the attestation process is not carefully implemented under realistic assumptions, the adversary may succeed. Therefore, even if a device comes up with a valid response, it does not mean that it has not been compromised. *Static* integrity measurement approaches are eminently vulnerable to ROP attacks since these attacks use the already existing code without altering it. Just as important as verifying that the code residing in static memory regions has not been modified, is to verify that the code is being executed as it was meant to be. This is the aim of *dynamic* integrity measurement approaches. However, due to the diversity and dynamicity of run-time properties, it is not an easy task to identify the known good states of dynamic objects [75].

2.2.3 *Timing*

In any practical implementation of an attestation mechanism, the verifier will only wait for a limited amount of time for a prover's response after sending a challenge. While some proposals have a *strict* timing condition, others adopt a more *loose* approach.

SoftWare-based ATTestation (SWATT) [44] is the first attestation mechanism designed specifically for embedded devices. It relies on *strict* timing of challenges and responses to detect compromised provers. The verifier sends the prover a randomly generated nonce that is used as the seed to the prover's Pseudo-Random Number Generator (PRNG). The prover then performs a cell-based pseudo-random memory traversal, iteratively reading memory words and computing a checksum of its program memory contents. Therefore, an adversary cannot predict the order of memory accesses, and if the memory has been altered, the attacker has to modify the attestation routine and insert statements checking whether the current address was modified. If that is the case, then, to get the right response, the adversary has to redirect the memory access to the memory location where the original value is. The authors' main assumption is that the attestation routine is constructed in a time-optimal way so that any modifications to it would result in a detectable increase in computation time which the verifier would detect. So, the verifier detects compromised provers either because the returned checksum is wrong, or because the response is delayed. However, Castelluccia et al. [24] presents an attack that is faster than the one the authors of SWATT considered. Furthermore, another possibility is to overclock the prover's CPU, such that, even if more instructions have to be executed, the total amount of time taken would still be within the limits. Although, this would be considered a hardware modification attack, and thus, assumed not to occur. Partly to address this, Kong et al. [76] proposed to incorporate the outputs of a PUF into the checksum computation to overcome overclocking, as well as impersonation attacks.

Choi et al. [77] propose a proactive code verification protocol for WSNs with *loose* timing conditions. In essence, the main idea is to fill the prover's memory so that an adversary has no place to hide its malicious code. The base station adopts the role of the verifier and is assumed to share a pairwise key with every node in the network. The prover receives a nonce from the base station and uses it as the seed to a Pseudo-random Function (PRF) whose outputs are used to fill empty memory regions. Afterwards, the prover calculates a hash over its memory and sends the result to the base station for verification. The issue with this approach is that the random contents used to fill the memory are being generated by a PRF executing on the prover. So once a node is compromised, the attacker has access to the PRF and can use it to calculate the hash on-the-fly, without ever storing its outputs in memory. Even if this takes more time than the normal protocol execution, the proposed scheme does not strictly control the execution time of the attestation routine, and the attack would pass undetected. A second issue is the use of cryptographic keys with no tamper-resistant hardware.

2.2.3.1 *Discussion*

Theoretically, the larger the time limit for a prover to respond to a challenge, the higher the number of attacks an adversary can explore. Whether an approach relies on accurate measurements of the attestation routine execution time depends on the system model and the assumptions made.

Approaches with *strict* timing conditions typically do not rely on tamper-resistant hardware and assume a time-optimal implementation of the attestation routine. Otherwise, an adversary could develop a faster routine and use the time saved to forge a valid result. However, the work by Castelluccia et al. [24] highlights three limitations of these approaches. Firstly, it is very difficult to correctly design a time-optimal attestation routine since its implementation must be small and simple. This precludes the use of cryptographic functions, which are complex and time consuming, and favors the use of simple instructions, such as *add* and *xor*, which may achieve poor security. Secondly, to achieve maximum speed, these routines are implemented in assembly, which is highly error-prone. Thirdly, proving the run-time optimality, for both the attestation routine and best possible attack, remains an open research problem. Another issue with these approaches is that they need to assume a maximum network round-trip time (RTT) so that they can define a timeout which allows all honest provers to reply in time. However, sensor nodes communicate over an unreliable wireless medium and can suffer from unpredictable faults and external interference. On the other hand, approaches that do not depend on accurate measurements of the execution time either rely on tamper-resistant hardware or make assumptions that limit the adversary capabilities. One of the assumptions made is that if all memories of the prover are filled, then an adversary has no space to allocate malware and still manage to compute a valid response. However, these assumptions do not always hold. For example, an adversary might compress the existing code in memory gaining enough space for the malware, or colluding nodes might install the malware in different memory locations, such that when they communicate they can recover the contents that were locally overwritten.

2.2.4 *Memory Traversal*

During attestation, a prover uses its memory contents as evidence of its trustworthiness. The prover has essentially two different ways to traverse its memory: *sequentially* or *pseudo-randomly* where the later can be further classified into *cell-based* or *block-based*.

Sequential memory traversal approaches go through a prover's memory in an iterative manner from start to end. For example, Park and Shin [33] present a soft tamper-proofing scheme for WSNs based on *Program Integrity Verification (PIV)*. The network is divided into clusters, operated by cluster heads, named PIV Servers (PIVSs), which have better computation and storage capabilities than normal sensors. PIVSs work as verifiers and for each attestation round they generate a different attestation routine, PIV Code (PIVC), which is executed by the prover. By using Randomized Hash Functions (RHFs), PIVSs randomly encode hash computation algorithms for each PIVC created. To protect sensors from adversaries impersonating PIVSs, there are multiple Authentication Servers (AS) deployed over the network. An AS works as a Trusted Third Party (TTP) allowing sensor nodes to confirm the authenticity of PIVSs and, consequently, the PIVC. The authors do not discuss the AS in further details stating they can either share a symmetric key with nodes or use public-key cryptography; though again the nodes are not assumed and are not likely to possess tamper-resistant hardware to secure cryptographic keys. Prior to deployment, a nodes' program memory is partitioned into multiple blocks. For each block, a digest is calculated and stored in a database accessible by all PIVSs. The digests are then classified into three categories: common to all nodes, common to a group of nodes, or unique to a specific node. Therefore, the number of digests to be stored can be greatly reduced. Before gaining access to the network, a node must prove its integrity. Thus, during the initialization phase all nodes ask to be verified. The RHF generates the same result if it takes either the program block or its corresponding digest as input. Consequently, both the prover and the PIVS are able to obtain the same result. Starting from the first address of the program memory towards the last, a checksum is computed for each memory block and a final checksum over these checksums represents the entire memory. The authors propose to initialize the data memory with random values that can neither be predicted nor compressed. However, an adversary could still compress the original software, residing at M_p, and calculate the digest for the compressed code blocks using on-the-fly decompression techniques.

Secure Code Update By Attestation (SCUBA) [42] is a mechanism for the detection and recovery of compromised nodes in WSNs. It relies on Indisputable Code Execution (ICE), which guarantees untampered code execution regardless of whether the node has been compromised or not. The approach requires each sensor node to have enough space in ROM to store its own ID and the base station's public key. Therefore, an adversary cannot modify these values even after it compromises a node. The approach is similar to SWATT and traverses the memory in a cell-based pseudo-random manner, however, it does not verify the entire program memory, and it extends the checksum computation to include dynamic data. Differently from Spinellis, ICE takes the CPU state — Program Counter (PC), Data Pointer (DP), and Status Register (SR) — as input to calculate the checksum.

To secure its untampered execution, the attestation routine disables all interrupts. Then it computes a checksum over the memory regions containing itself, the SCUBA protocol executable, the node ID and base station's public key, as well as the CPU state. It sends the result to the base station and starts the SCUBA executable, with interrupts still disabled, guaranteeing it executes untampered. As with SWATT, the attestation routine is constructed in such a way that the checksum will either be incorrect or its computation will be notably longer if the routine is modified. In such cases, the base station can assume the node has been compromised and blacklist it. Otherwise, the SCUBA protocol can further verify the node, inspecting and repairing the rest of its memory. A downside of the proposed scheme is that the PC may not be accessible depending on the platform. Furthermore, Castelluccia et al. [24] describe an attack that overcomes ICE's checksum computation and is able to execute arbitrary code without being detected. The attack takes advantage of the fact that the additions performed by the ICE checksum function discard the carry. Therefore, changes in the most significant bit (MSB) may not alter the checksum result. This allows an adversary to store a copy of the ICE function in a different memory position, such that its address only differs from the original location in its MSBs.

2.2.4.1 *Discussion*

Sequential memory traversal approaches are simple to implement and efficient — they run in linear time according to the memory size. Furthermore, they provide complete coverage over the memory regions being verified, passing through each memory address a single time. Nevertheless, these advantages come at the cost of a disadvantage, which is predictability. The fact that memory addresses are checked only a single time and in a predictable order makes it easier for adversaries to counterfeit attestation. For instance, an adversary can move memory contents around to avoid detection. When verification starts, malware can be moved to the end of the memory and right after the verification passes through its original position the malware can be moved back. Another possibility is for two colluding nodes to install the malware in different memory locations, and when one of them is under attestation, it asks the other for the contents it has overwritten. In contrast, pseudo-random approaches, are intrinsically unpredictable, and not victims of the same attacks. However, they are less efficient and provide only a probabilistic guarantee of memory coverage. To ensure, with high probability, that each memory address is accessed at least once, these schemes rely on the result of the Coupon Collector's Problem [78], which states that for a memory of size n it is necessary to perform $O(n \ln n)$ memory reads. Consequently, some memory addresses end up being accessed multiple times introducing unnecessary overhead. To reduce this overhead, Yang et al. [39] proposed to traverse the memory in a block by block manner. Rather than accessing the memory one address at a time, block-based approaches access blocks of addresses and perform *xor* operations within blocks. For a block size b, $O(\frac{n \ln n}{b})$ iterations are necessary to cover each memory address at least once. It is interesting to see that when $b = 1$ the scheme functions as cell-based approaches, and when $b = n$ it works as a sequential traversal. Choosing the size of b is, therefore, a tradeoff between performance and security. Furthermore, as observed by Armknecht et al. [29], if the block size is determined prior to attestation, an adversary can perform collision attacks against block-based schemes. These are forgery attacks that work by altering addresses inside a block in such a way that modifications neutralize one another.

2.2.5 *Attestation Routine*

Most existing approaches have their attestation routine *embedded* in the prover's memory prior to the network's deployment. However, a verifier may also generate and send the prover different routines for each attestation round.

For example, Shaneck et al. [79] propose a remote software-based attestation framework were the attestation routine is generated *on-the-fly*. The base station plays the role of the verifier and is assumed to be within communication range of all network nodes. Furthermore, it shares a symmetric key with each node to secure communication. At each attestation round the base station generates a new attestation routine and sends it to the prover. It waits for a maximum time that comprises the time to send and receive a message, the attestation routine execution time, and the expected network delay. The procedure utilizes techniques such as randomization, encryption, obfuscation, and self-modifying code to prevent an adversary from avoiding detection. They propose to use random keys to encrypt the entire attestation routine and send a corresponding decryption routine together with the code. This routine is also responsible for discovering the key, which is located somewhere in the prover's memory, hidden through opaque predicates [80]. The attestation routine has three main components: seed calculation, memory reads, and hash computation. The first component is responsible for initializing the PRNG, which determines the order of memory accesses. So it is in the interest of an adversary not to modify this part, but to infer the seed value. However, the seed computation also uses opaque predicates. The other two components are part of a loop, that reads memory addresses and use their contents to compute the hash. As with SWATT, it is necessary to iterate through the loop several times to cover the program memory. The second component is the one an adversary would attempt to modify so that it could redirect memory reads. To avoid such modifications this component has several junk instructions, which appear to be reachable due to opaque predicates, and it self-modifies its code relocating the read instruction at each iteration. After the hash calculation is complete, the result is sent to the base station. The authors have neither implemented nor evaluated their proposal, so it is difficult to discuss its security and even its feasibility. One difficulty to implement this proposal is that several commodity sensor nodes store the executable code in flash memories programmable only by pages. Another issue is the use of cryptographic keys to secure communication, which, once again, is made without considering that commodity nodes do not have tamper-resistant hardware to protect them from attackers.

2.2.5.1 *Discussion*

Approaches that embed the attestation routine in the prover's memory provide security by design. The attestation routine is conceived to be secure, such that a prover will fail, with high probability, to provide a valid response if either it changes the routine or the memory contents under verification. On the other hand, approaches that generate the attestation routine on-the-fly provide security through obscurity. Since a new routine is generated for each attestation round, attackers cannot predict what instructions should be executed and even less the outcome. The attestation routine may actually have vulnerabilities, but since their implementation is hidden from opponents, these vulnerabilities are unlikely to be explored in a timely manner.

2.2.6 *Program Memory*

It is possible that the program of a sensor node does not occupy its entire program memory, thus leaving empty spaces. An adversary can, therefore, take advantage of this space to store data used to overcome attestation. To avoid this, some approaches propose to *fill* the empty space with incompressible random noise.

For example, AbuHmed et al. [34] introduce two software-based remote code attestation procedures for WSNs. The authors also consider a scenario where the base station acts as verifier and shares cryptographic keys with sensor nodes without any tamper-resistant hardware. In this context they present two techniques, one pre- and one post-deployment, to fill the empty memory space of sensor nodes with incompressible random noise. In the pre-deployment approach, the program memory of each node is filled using a seed and the verifier keeps a register of the seed together with the corresponding node ID. Then, during attestation time, the verifier can generate a copy of the node's memory to compute the checksum and compare the result with the one received. In the post-deployment approach, a node uses some dynamic data gathered from the environment as the seed to generate the noise and fill its memory. After that, the sensor transmits the seed to the verifier and deletes it from its memory. Consequently, an adversary who compromises the node afterwards has no space to store its malicious code, and if it does overwrite the memory, it cannot regenerate the original contents without the seed. The authors also propose two block-based memory traversal algorithms with variable block sizes, in contrast to Yang et al. [39] where the size is always the same. In the first algorithm, the verifier defines the size of the block together with the challenge. In the second algorithm, a dynamic block size that changes during the checksum computation is used. However, both schemes are still vulnerable to compression attacks, as an adversary can compress the original software and calculate the digest for the compressed code blocks using on-the-fly decompression techniques.

2.2.6.1 *Discussion*

Physical memory contents are typically of low entropy and thus compressible [81]. As a result, even if empty spaces are filled with incompressible random noise it may still be possible for an attacker to compress the original program code and gain enough space for its malicious code [24]. To defend against this attack, Vetter and Westhoff [16] present a code attestation mechanism with compressed instruction code. Each sensor node is uploaded with a compressed code image, and its remaining program memory is filled with incompressible random noise. The code image is divided into blocks of equal length which are individually compressed. However, after compression these blocks may not have the same size. Therefore, the program memory is divided into two sections, one to store the compressed code blocks and another one, named Line Address Table (LAT), to store the block's offsets. In order to execute the compressed code, they incorporate a hardware extension: a dedicated microcontroller uses the LAT section to decompress code blocks, and a cache, maintained within the node's RAM, is used to store the decompressed code block under execution. They implemented the same attestation routine used by SWATT [44], however, they do not have the same strict timing conditions since the memory is completely full. Nonetheless, the data memory is not verified, and an attacker can perform ROP attacks or even modify the decompressed code held in the cache to compromise a node. Besides, the need for a dedicated microcontroller to decompress code blocks is a strong limiting factor for the applicability of this scheme.

2.2.7 *Data Memory*

Depending on the prover's memory architecture, the data and program memory physical addresses can either be shared with (von Neumann architecture) or separated from (Harvard architecture) one another. In the latter case, it is common for the size of the program memory to be much bigger than the data memory. Therefore, an attacker exploring the data memory has only a limited amount of space [79]. Also, in the Harvard architecture, the contents of the data memory are not executable. For these reasons, some approaches do not verify the data memory. However, attacks exploring the data memory have already been demonstrated. For example, the work by Castelluccia et al. [24] describes a rootkit-based attack that hides malicious code in the data memory during attestation.

Some *dynamic* attestation mechanisms try to *verify* the data memory. For example *Dataguard* [32] is a software attestation scheme for dynamic data integrity based on *data boundary integrity*. Each node has a unique seed, which is erased after initialization, that is used to insert data guards, similar to canary words [82], around data objects. When challenged by a verifier the prover sends a response based on the values of all data guards. Therefore, if an adversary overwrites a data guard, by performing a buffer overflow attack, for example, it will not be able to recover the original data guard value since it no longer has the seed. However, this scheme also has its own limitations. Firstly, the method is vulnerable to attacks that do not modify data guards. Secondly, it only provides a coarse-grained data protection where each memory block is treated as a unique object. Consequently, the scheme does not protect individual array elements.

Other approaches, such as the works of Park and Shin [33] and Perito and Tsudik [54] overwrite all contents of the data memory, erasing any malicious data in the process.

2.2.7.1 *Discussion*

Approaches that do not verify the data memory are inherently vulnerable to ROP attacks. However, existing approaches that verify the data memory are not completely safe either. Due to the difficulty to predict the behaviour of dynamic data, existing approaches only partially cover the data memory. While erasing all the data memory is the safer approach, it is not really attesting the memory contents. Furthermore, together with any malicious data, these approaches also eliminate all data a node has worked to achieve prior to attestation.

2.2.8 *Interaction Pattern*

Most existing approaches interact in a *one-to-one* pattern, as depicted in Figure 1, where for each attestation round there is one verifier and one prover. However, this is not the only possible way to perform attestation.

For example, Jin et al. [15] propose an *Unpredictable Software-based Attestation Solution (USAS)* to detect compromised nodes in mobile WSNs that uses a *one-to-many* interaction pattern by creating dynamic attestation chains. Each chain comprises a single Initiator (I-node) and several Follower nodes (F-nodes). In each attestation round the base station, acting as the verifier, challenges a randomly selected I-node. The challenge consists of a random number, which is used as input for the attestation routine, and authentication messages for the I-node and the F-nodes. When challenged, the I-node runs the attestation routine and uses its output to challenge the F-nodes, which then execute the attestation routine and send the result back to the base station. Prior to the network deployment, each node has its program memory filled with pseudo-random noise and the base station keeps a copy of the seeds used for all nodes. The base station then compares the results from the F-nodes with the expected ones to detect compromised nodes. As long as one

F-node result is valid the I-node can be considered genuine as well. However, if all F-nodes return invalid results, nothing can be said about all attested nodes. In this case, either all F-nodes have been compromised, or the I-node has been. Performing all attestation chain, which is not a cheap operation, and not being able to affirm anything is a significant limitation of this scheme. Another limitation of this proposal is that, even though it creates an attestation chain, it attests only sensors that can directly communicate with the base station to avoid intermediate nodes tampering with the messages.

Another possibility is to orchestrate a distributed attestation, in a *many-to-one* interaction pattern, where the neighbors of a node work together to attest it. For example, Yang et al. [39] present two distributed software-based attestation schemes where sensor nodes collaborate with each other to attest the integrity of their neighbors. The main difference from previous approaches is that only regular nodes are involved, and this approach does not require trusted verifiers. The authors propose to fill the empty spaces in each sensor's program memory with pseudo-random numbers, but differently from Choi et al. [77], this is done prior to node deployment. For each node, a different seed is used for generating the pseudo-random numbers. After deployment, every sensor discovers and establishes a pairwise key with each of its neighbors — again, cryptographic keys are used with no protection. In the first proposed scheme, a node splits its seed into multiple shares, sends a separate share to each neighbor, and then deletes the seed from memory. An attestation is triggered when more than half of a node's neighbors detect its abnormal behavior. In this case, all neighbors elect a cluster head, which is different for each attestation round. The cluster head challenges the node, with a random number, and collects the seed shares from other neighbors to recover the seed. Then it locally computes the expected result and compares it with the response from the challenged node. The authors propose a new method to randomly traverse the program memory: instead of reading one memory word at a time, as was done in SWATT, they read a block of memory addresses and perform *xor* operations within blocks at each iteration. By using an appropriate block size, they are able to reduce the total amount of iterations while still covering the whole memory. In the second scheme, every node is also preloaded with tuples of challenge and response to its own M_p. Instead of sending shares of the seed, a node sends a tuple to each of its neighbors and then deletes all the tuples. When attestation is triggered, each neighbor attests the node sequentially using the challenge and response from the received tuple. A node is considered to be compromised if it does not pass the majority of attestations. A limitation of these approaches is that they require a minimum network density to work. If a node does not have enough neighbors, then it cannot be properly attested. Furthermore, both schemes incur considerable communication overhead. The first is more vulnerable to compromised nodes as they can send forged shares of the seed, or even worse, be elected cluster head. In the second scheme, an adversary would have to compromise more than half of neighboring nodes to succeed. However, this latter scheme requires the prover to execute the attestation routine once for each of its neighbors, which is both time and energy consuming.

2.2.8.1 *Discussion*

Simplicity is the main advantage of the one-to-one interaction pattern. It allows the verifier to target each node of the network individually. However, if several nodes need to be attested, the overall time increases linearly with the number of nodes. Attesting nodes in a one-to-many pattern reduces the overall computation time by allowing multiple nodes to execute the attestation routine in parallel. However, if an attestation chain is used, as in the works of Jin et al. [15] and Asokan et al. [20], a compromised node may discredit the entire chain. An important characteristic of both one-to-one and one-to-many approaches is that the verifier must be a trusted entity. This has significant implications because some proposed attestation mechanisms require the prover to be in the verifier's communication range. Therefore, a mobile verifier or multiple verifiers scattered across the network are necessary. Moreover, a verifier becomes a single point of failure, if only a single one exists. The main feature of the many-to-one pattern is that it can be performed without the use of trusted verifiers. This has both advantages and disadvantages. The upside is that it allows several nodes to be attested at the same time in a distributed manner without having a single point of failure. Besides, it closes the gap between detection of misbehaviour and attestation, since neighboring nodes are the direct observers of a compromised node. The downside is that it requires a minimum network density and is more vulnerable to compromised nodes as they can not only attempt to avoid detection but can also mislead the outcome of other nodes' attestation.

2.2.9 *Summary*

Having examined all characteristics and their instances, one can realize that there is no definitive attestation mechanism. Each approach has its own advantages and disadvantages. Choosing the best solution for a specific scenario depends on a series of factors, such as the assumptions made about the adversary, the environment in which the nodes are going to be deployed, how the sensors are going to be placed across it, and the underlying nodes hardware. Nevertheless, the discussion held in this section helps to clarify the tradeoffs between different techniques and can serve as a guideline to anyone choosing an existing attestation mechanism or designing a new one.

Table 2 maps, in chronological order, a representative number of attestation mechanisms to the taxonomy illustrated in Figure 3. To the best of our knowledge, there are no existing mechanisms that would not fit on the proposed taxonomy. Furthermore, we believe our taxonomy still holds outside the WSNs scenario. One important note is that we classify the mechanisms as they are first described in their original manuscripts, without considering proposed extensions. The main observation that we make is that even after compression and ROP attacks have been demonstrated [24] new static approaches were proposed [15, 16, 17, 18, 19, 20, 21] that completely ignored or did not provide sufficient protection against such attacks. We believe that an attestation mechanism should reflect the actual running system, and not just some static part of the prover's memory.

2.3 OPEN RESEARCH PROBLEMS

Existing attestation mechanisms are far from perfect, with much room for improvement. In this section, we examine open research problems and give directions on how to undertake them. We focus on four main topics, which we believe are of great importance and can receive further attention.

2.3.1 *Overoptimistic Assumptions*

Problems do not cease to exist just because they were assumed not to occur. Many attestation mechanisms rely on strong suppositions that may not hold in practice which may hinder their deployment. Moreover, this creates a gap between assumptions made and real adversary capabilities that may lead to security vulnerabilities. For instance, schemes such as SWATT [44] and SCUBA [42] assume a time-optimal attestation routine and use a strict timing condition which relies on this premise. Despite recent advances in code optimization [85, 86, 87], proving code optimality is still an open problem. In this case, one can draw a parallel with cryptosystems which rely on unproven assumptions, e.g. the RSA problem [88]. By making the algorithm public, under the scrutiny of a large community, one can have a higher confidence in the security achieved.

An additional assumption done by software-based attestation mechanisms with strict timing conditions is that using the maximum known network RTT to define the attestation timeout allows all honest devices to reply in time. While this assumption usually holds in controlled laboratory environments, it is typically untrue in real deployments and particularly so in the context of WSNs where many devices communicate over an intrinsically unreliable wireless medium and can suffer from issues such as external interference. Furthermore, the higher the attestation timeout, the higher the number of computations the untrusted device has to perform, consuming extra time and energy and restraining

Table 2: Mapping of some existing attestation mechanisms to the proposed taxonomy.

Approach	Evidence Acquisition	Integrity Measurement	Timing	Memory Traversal	Attestation Routine	Program Memory	Data Memory	Interaction Pattern
Reflection [31]	Software-based	Static	Loose	Sequential	Embedded	Unfilled	Unverified	One-to-one
SWATT [44]	Software-based	Static	Strict	Cell-based pseudo-random	Embedded	Unfilled	Unverified	One-to-one
PIV [33]	Software-based	Dynamic	Loose	Sequential	On-the-fly	Unfilled	Erased	One-to-one
Self-modifying code [79]	Software-based	Static	Loose	Cell-based pseudo-random	On-the-fly	Unfilled	Unverified	One-to-one
SCUBA [42]	Hybrid	Dynamic	Strict	Cell-based pseudo-random	Embedded	Unfilled	Unverified	One-to-one
Proactive [77]	Software-based	Static	Loose	Sequential	Embedded	Filled	Unverified	One-to-one
Distributed [39]	Software-based	Static	Loose	Block-based pseudo-random	Embedded	Filled	Unverified	Many-to-one
SAKE [83]	Hybrid	Dynamic	Strict	Cell-based pseudo-random	Embedded	Unfilled	Unverified	One-to-one
Memory filling [34]	Software-based	Static	Loose	Block-based pseudo-random	Embedded	Filled	Unverified	One-to-one
ReDAS [75]	Hardware-based	Dynamic	Loose	Sequential	Embedded	Unfilled	Verified	One-to-one
USAS [15]	Software-based	Static	Loose	Cell-based pseudo-random	Embedded	Filled	Unverified	One-to-many
PoSE [54]	Hybrid	Dynamic	Loose	Sequential	Embedded	Filled	Erased	One-to-one
DataGuard [32]	Software-based	Dynamic	Loose	Sequential	Embedded	Unfilled	Verified	One-to-one
SMART [67]	Hybrid	Dynamic	Loose	Sequential	Embedded	Unfilled	Verified	One-to-one
Compression [16]	Hybrid	Static	Loose	Cell-based pseudo-random	Embedded	Filled	Unverified	One-to-one
Lightweight [17]	Software-based	Static	Loose	Block-based pseudo-random	Embedded	Filled	Unverified	One-to-one
TrustLite [68]	Hybrid	Dynamic	Loose	Sequential	Embedded	Unfilled	Verified	One-to-one
PUFatt [76]	Hardware-based	Dynamic	Strict	Cell-based pseudo-random	Embedded	Unfilled	Unverified	One-to-one
TRAP [18]	Hardware-based	Static	Loose	Sequential	Embedded	Unfilled	Unverified	One-to-one
LRMA [19]	Software-based	Static	Loose	Cell-based pseudo-random	Embedded	Unfilled	Unverified	One-to-one
SEDA [20]	Hybrid	Static	Loose	Sequential	Embedded	Unfilled	Unverified	One-to-many
C-FLAT [84]	Hardware-based	Dynamic	Loose	Sequential	Embedded	Unfilled	Verified	One-to-one
DARPA [21]	Hybrid	Static	Loose	Sequential	Embedded	Unfilled	Erased	One-to-many
AAoT [23]	Software-based	Dynamic	Loose	Block-based pseudo-random	Embedded	Filled	Unverified	One-to-one

it from performing its primary tasks. Therefore, a mechanism that takes into account the actual network conditions is needed.

Other approaches assume that filling the empty spaces of a prover's program memory leaves the adversary with no space to store its malicious code without being detected. Yet, only two of these approaches [54, 16] are capable of defending against code compression attacks; and they have their own limitations. Perito and Tsudik [54] require all the prover's writable memories to be overwritten while Vetter and Westhoff [16] require a dedicated microcontroller.

Another type of attack that most approaches assume does not happen or impose restrictions to their occurrence are collusion attacks. Although it is possible to imagine solutions to these attacks, such as jamming the prover or having its neighbors monitor its communication during attestation, the feasibility, effectiveness, and impact of these solutions have not been analysed.

Several software-based proposals assume a secure and authentic communication channel between prover and verifier through the use of cryptographic keys defined prior to network deployment. However, the nodes used in these approaches do not have the necessary hardware to protect these keys; and, oddly enough, the absence of such hardware is the very motivation behind their development. Cryptographic keys can, however, be used on commodity sensor nodes when determined at run-time. For example, Software Attestation for Key Establishment (SAKE) [83] is a protocol for establishing shared keys between two neighbouring nodes without assuming prior authentic or secret information. However, SAKE is based on ICE, the same primitive used by SCUBA, which relies on strict time measurements. More recently, Feng et al. [23] proposed a lightweight Attestation and Authentication of Things (AAoT) scheme that uses the microcontroller SRAM as a PUF to provide authentication without requiring any hardware changes. Another alternative solution to authenticate sensor nodes is the use of radio frequency fingerprint techniques which enable the identification of individual devices by the unique characteristics of their radio transmitter [89, 90, 91].

The vast majority of attestation mechanisms do not pay much attention to the prover's security. They are developed under the assumption that the verifier can always be trusted. However, an adversary can take advantage of this and impersonate the verifier to perform DoS attacks targeting honest devices. Brasser et al. [92] investigate attacks and countermeasures under this context. They describe three attacks an adversary may explore: replay, reorder and delay of attestation challenges. The countermeasures involve the use of a secret key for authenticating the verifier and the use of either a challenge counter or timestamp (which requires a synchronized clock between prover and verifier), all of which need hardware protection.

While it is impractical for an adversary to launch a large scale attack that requires physical access to compromise a sensor node, it may be enough for an adversary to compromise just a few selected nodes in this manner. Nevertheless, the vast majority of software-based and hybrid approaches do not take into account physical attacks. As an alternative, Ibrahim et al. [21] proposed Device Attestation Resilient to Physical Attacks (DARPA). Assuming that to perform a physical attack an adversary has to capture and temporarily disable the target sensor for a perceptible amount of time; DARPA requires all nodes to periodically broadcast a message that serves to prove the node is active. All nodes log these messages which can then be collected by the verifier during attestation. While the scheme may suffer from false positives because of device or network failures, it is a first step towards detecting physical attacks.

2.3.2 *Effectiveness*

WSNs are usually constituted of resource constrained nodes that have limited energy, low processing power, and little storage space. In many application scenarios, the network needs to operate unattended for long periods of time, so battery depletion is crucial. So whilst ensuring the integrity of the sensor network is important, it is equally important that security mechanisms should not have a high impact on the system performance. Attestation procedures typically degrade both network throughput and the node's battery lifetime. Furthermore, attestation only allows to ascertain the state of the prover at a particular point in time. How often should then attestation be performed to provide assurance of the network's integrity whilst not depleting resources?

Chen et al. [93] present a model to analyse the frequency with which code attestation should be performed to maximize sensors lifetime while effectively detecting compromised nodes. However, their model considers solely designs in which attestation is invoked probabilistically and ignores aspects such as using network monitoring information to decide when to trigger attestation next. Their model leads then to the rather straightforward conclusion that the frequency at which attestation must be performed depends directly on the rate at which nodes can be compromised (*compromise rate*). In practice, the compromise rate is unknown a priori and is highly dependent on many different factors such as the network deployment, types of attack, and value and timeliness of the information sensed by the network. For example, it has been shown that malicious packets can propagate quickly and compromise entire networks in short periods of time [8, 94]. However, in their analysis, Chen et al. ignore such aspects and assume a compromise rate in the range of 0.0058 to 0.0072 nodes per hour, corresponding to a node being compromised once every five to eight days.

Further analysis is needed to determine not only when to perform attestation, but also which devices to attest and what to do when a compromised node is detected. For example, nodes closer to the base station may have a higher value as targets since they also forward much of the data from peripheral nodes to the base station and vice-versa. Such aspects should be taken into account, e.g., through different weights, when deciding which nodes to attest. Furthermore, a compromised node can easily broadcast malicious packets to its neighbors. So attesting the neighbours of a compromised sensor would be an effective way of reducing the adversary compromise rate and limit diffusion.

2.3.3 *Time of Check to Time of Use*

Perhaps the biggest issue when using attestation in WSNs is the gap between the Time of Check to Time of Use (TOCTOU) [95]. Attesting a node occurs at a particular point in time and does not guarantee that the node has not been temporarily compromised before, or that it will not be compromised right after attestation. An adversary can perform a TOCTOU attack as long as it has some unmeasured location to hide its malicious data before attestation starts and is able to reinstall it after attestation ends [96]. In fact, Castelluccia et al. [24] present a rootkit-based attack that does exactly this. The difficulty arises from attesting all the memories of a prover. Some dynamic attestation approaches [75, 32] try to close the TOCTOU gap by checking run-time properties of the software execution. However, it is not always possible to predict the behaviour of dynamic data, and these approaches end up covering only a subset of the required properties. ICE [42] attempts to guarantee an untampered execution environment to ensure that a piece of code runs unmodified. The limitation of this approach is that it applies solely to self-contained code that does not invoke other software on the prover and executes with interrupts disabled. Another approach to reduce the TOCTOU gap could be the integration of attestation and Control Flow Integrity (CFI) [97] techniques. Protecting the control flow of a program prevents adversaries from arbitrarily altering its execution. CFI techniques achieve this protection by embedding control code in the original application program. Attesting that both control and application code have not been altered is a further indication that the original program is being executed as it was supposed to. Ferguson and Gu [98] implement control flow protection in the context of WSNs; however, the described scheme has not been combined with any attestation mechanism. Control-Flow ATtestation (C-FLAT) [84] is a similar, but different, approach. Instead of embedding code to protect the authentic application flow, it embeds code to monitor its execution path. This allows the verifier to attest the prover's run-time behaviour, detecting any control flow diversion. Nonetheless, it is important to note, that CFI is not infallible [99] and does not ensure data flow security [100, 101, 102].

Significant effort has been dedicated recently to approaches capable of defending against ROP attacks. These broadly fall into two categories: one, is the already mentioned CFI, while the other is software diversification [103]. However, up to now, all proposed mechanisms can be subverted in some way. Therefore, ideas continue to evolve, and new approaches continue to emerge. A promising countermeasure to such attacks is the use of execute-only memory [104, 105], which allows marking pieces of memory as executable but not readable. However, in order to attest a device, the attestation routine needs to read the code it executes, and, thus far, no approach integrates the two techniques.

2.3.4 *Scalability*

To attest a device, a verifier must know its expected internal state and hardware architecture. In a simple WSN scenario, all the nodes may be executing the same application on the same hardware platform, making this reasonably easy. However, even in such cases, scalability can become an issue of effectiveness since the time and energy required to attest all nodes can be prohibitive depending on the network size. Furthermore, in many other situations, the network is typically heterogeneous and comprises different types of sensors, or different cluster nodes execute different applications. In more extreme cases, the network may contain many sensors from distinct vendors executing several different applications. How can one then store and manage the information necessary to attest these devices in a scalable manner? The approach proposed by Park and Shin [33] partitions the application code into multiple blocks, calculates a digest for each block and classifies the block according to whether it belongs to all, a group, or a unique node. Although this method allows to significantly reduce the amount of information needed, it also has its own security vulnerabilities. For instance, an adversary may compute the prover's digests before modifying its memory and then only keep a copy of the digests to pass attestation. Another approach is to have similar devices, which execute the same application, attesting each other in a distributed fashion. However, to be viable, this approach requires a minimum density of similar devices scattered across the network.

Software updates further complicate the issue. The slightest change in the software a prover executes can impact the attestation outcome. Therefore, additional information needs to be maintained to know which devices are running which versions. Sadeghi and Stble [106] propose to attest a device based on the properties that it offers instead of its software and hardware components. However, in most scenarios, property-based attestation requires the use of a trusted third party to map the device's components to properties [107]. Moreover, the identification and formal definition of security properties are still open problems [108].

A more recent approach, named Scalable Embedded Device Attestation (SEDA) [20], was designed with the main goal of attesting a large amount of devices. The most remarkable aspect of SEDA is that the verifier does not need to know the detailed configuration of all provers. However, to achieve this, the approach relies on cryptographic secrets that must be secured by hardware. With these secrets, the nodes can be issued a certificate of its valid software configuration, during initialization or after an update, and can share the certificate with their neighbors. Attestation is performed in a one-to-many interaction pattern, where each device attests its neighbors and reports back to its parent. As stated in Section 2.2.8, the main limitation of this interaction pattern is that a compromised node invalidates the results for all the nodes following it on the attestation chain. In the worst case scenario, the first node in the chain has been compromised, and resources used to perform attestation are discarded. Another limitation of this approach is that it only provides a static integrity measurement.

2.4 CONCLUSIONS

Attestation is a critical service that enables the detection of compromised devices. However, there is no consensus on how to perform attestation in an effective and secure manner. Consequently, several mechanisms have been proposed over the last years. They differ not only in design choices and implementation but also in the assumptions they make over the system and adversary models. As we have shown in this chapter, some assumptions are more realistic than others in the context of WSNs, and not all proposals can be considered secure. Furthermore, we presented a taxonomy that identifies the main characteristics of proposed solutions and surveyed the state of the art mapping existing approaches to our taxonomy. This allowed us to discuss the tradeoffs between design choices made by different proposals. Finally, we have identified open research issues and given directions on how to tackle them.

Although we have centered the discussion around WSNs, we believe our taxonomy is still relevant outside this context. Moreover, all the analysis of advantages and disadvantages of different techniques done in this chapter can be directly applied to many Internet of Things applications, since they will either use sensor networks or share many characteristics in common. We hope this work serves both as a reference to avoid already committed mistakes and as a development guideline to future attestation mechanisms.

In the following chapters, we will address some of the open research issues we have identified, namely *effectiveness* and *scalability* in Chapter 3, and a critical *overoptimistic assumption* made by software-based attestation mechanisms with strict timing conditions in Chapter 4.

3 COMBINING ATTESTATION WITH MEASUREMENTS INSPECTION

As seen in Chapter 2, the goal of attestation mechanisms is to ascertain whether or not a device is compromised. However, attesting a device comes at a price: in order to be able to receive an attestation challenge, sensor nodes must stay with their radios on in listening mode; once a challenge is received they must perform the necessary computations; and finally, there is a cost to transmit the response back to the verifier. Nevertheless, these mechanisms do not indicate which and when or how often devices should be attested. As an adversary can attack any network node, given no auxiliary information, all nodes must be verified. The higher the attestation frequency, the earlier a compromise can be detected. Still, this frequency has a direct impact on the network's lifetime. Thus, there is a strict trade-off between the security level achieved and the cost to obtain it.

To improve this trade-off, we propose the use of attestation in combination with another security technique: measurements inspection. While attestation determines the integrity of each node individually by inquiring its memory contents, measurements inspection can detect malicious nodes by examining internal correlation structures between the measurements sent by different nodes. When performed in a centralized way, for instance by the base station, measurements inspection requires no additional computations from the sensor nodes themselves. From this perspective, it can be considered very lightweight, as opposed to attestation. However, the accuracy in distinguishing genuine from compromised nodes is limited by the unpredictability of the sensed phenomenon, which introduces uncertainty in the measurements correlations, and is also undermined when malicious nodes collude. Therefore, our goal is to combine both techniques to maximise the benefits of their complementary capabilities. By using measurements inspection to trigger attestation when something is suspicious we can keep the high reliability level of attestation and the low cost of measurements inspection. Nevertheless, measurements inspection is a multistep process, which means there are different ways to integrate both techniques, and it is not obvious which combinations are better beforehand.

In this chapter, we present a study on how to best combine attestation and measurements inspection. We first give an overview of how measurements inspection functions in Section 3.1. After that we define the performance and compare the advantages and disadvantages of both approaches in Section 3.2. In Section 3.3 we specify our assumptions and adversary model. Then, in Section 3.4 we introduce different combination strategies and their effect on the detection of malicious sensor nodes. In Section 3.5 we analyse the detection performance of each scheme in terms of accuracy and attestation frequency, which is the main factor to affect the energy consumption. Following this analysis, in Section 3.6 we describe our simulation experiments that model the whole data generation and transmission process in the WSN to obtain an accurate detection performance and energy consumption. In Section 3.7 we outline how our approach contrasts with related work. Lastly, we give our conclusions and possible directions for future work in Section 3.8.

The work presented in this chapter was done in collaboration with Dr. Vittorio Paolo Illiano at the time he was a colleague Ph.D. student at Imperial College London. Given his expertise, Dr. Illiano managed specific issues regarding measurements inspection, while I handled issues concerning attestation. Together, we designed the combination schemes, built analyses, and implemented the simulations that follow.

3.1 MEASUREMENTS INSPECTION

Measurements inspection refers to the detection of malicious data injections through analysis of the measurements themselves. The idea is that to disrupt the information gathered from the measurements, the injected and genuine values should not hold some internal data properties introduced by *inter-measurements correlation*.

Correlations occur in time, space, and between different sensed phenomena. In this work, we focus on spatial correlations, defined as the relationships existing between measurements at different points in space because in this context remaining undetected requires compromising more sensors. Hence, the first step of measurements inspection is a test for anomalies in the measurements correlation, i.e., their *detection*. However, if malicious nodes *collude*, i.e., act in concert according to a common goal, malicious measurements may be correlated and more difficult to detect. Moreover, even if the presence of malicious measurements can be detected, colluding nodes could make genuine sensors appear responsible for it. Thus, besides detection of malicious interference, the system also requires a *characterisation* step, to identify the sensors responsible for it. Finally, faulty sensors may also disrupt inter-measurements correlations and need to be distinguished from malicious ones. To make such distinction, a further step, known as *diagnosis*, is required. Nevertheless, this step is missing in most measurements inspection algorithms [109]. Figure 4 illustrates the high-level architecture of a complete measurements inspection scheme.

The *detection* step searches for anomalies, identified as measurements that introduce an anomalous variation when compared to the measurements of other sensors. In particular, the anomaly is examined in the individual variation that a measurement introduces in a small neighbourhood, contextualised into the overall variation that is observable in larger neighbourhoods. So, the detection step does not impose any constraint to the reported values but imposes a threshold on the variations at the low scale with respect to the higher scales events. Thus, remaining undetected (i.e., below the detection threshold) prevents the attacker from causing high damage (e.g., spoof a fire) and vice versa. The characteristics of the physical phenomenon, the deployment, and the environment, determine the cross-scale relationships, which are learnt and tested through the analysis of *wavelet* coefficients at multiple scales [110].

Characterisation identifies malicious sensors through a group-wise analysis that considers correlated measurements together, where correlations are still based on their effect on the neighbouring measurements. For instance, if two or more measurements are consistently

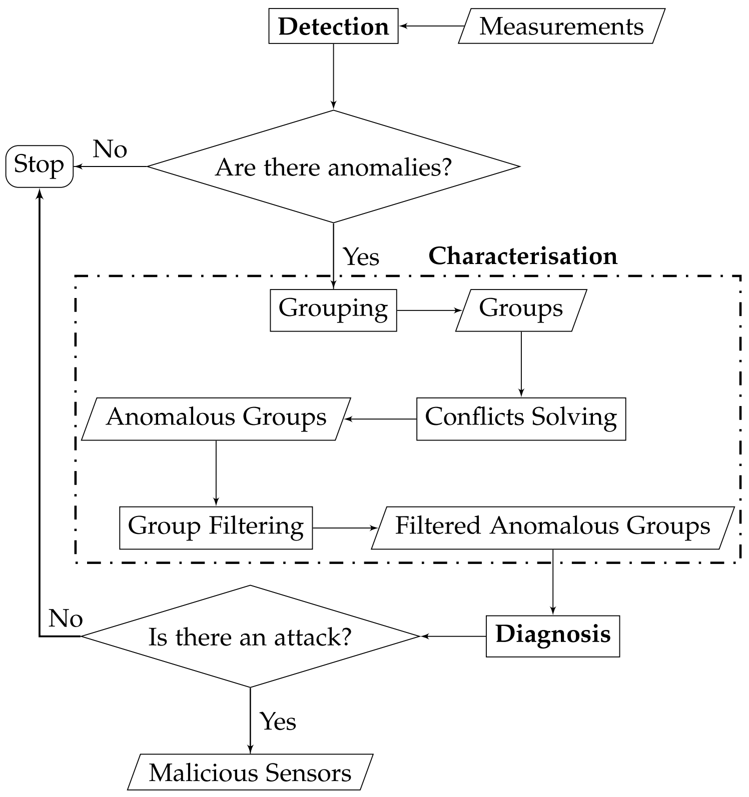


Figure 4: Measurements Inspection Scheme.

showing an increasing trend, they will be assigned to the same group. Consequently, colluding sensors are presumably assigned to the same group, while each set of genuine correlated sensors are assigned to different groups. Sensors are grouped together if they are within the area characterised by the same spatial behaviour. The anomalies observed in the detection step are then used to identify conflicts between groups, which reveal which group is responsible for the anomaly. A conflict solving algorithm finds the anomalous groups. The last characterisation task is to filter out sensors in the same group that have a borderline behaviour, i.e., their measurements neither endorse nor reject the group's spatial behaviour.

Finally, by searching for characteristics typical of faulty sensors in the anomalous measurements, the *diagnosis* step can infer whether each anomalous sensor is likely faulty or malicious.

Table 3: Notation summary.

Term	Description
N	Number of sensor nodes
C	Number of malicious sensor nodes
A_{D_T}	Anomaly Detection triggers
$A_{C^S_F}$	Sensor S fails Anomaly-based Characterisation
$A_{D^S_M}$	Anomaly-based Diagnosis output for sensor S is "Malicious"
MI^S_F	Sensor S fails Measurements Inspection
$A_{T^S_F}$	Sensor S fails attestation
$A_{T^S_P}$	Sensor S passes attestation
S_G	Sensor S is genuine
S_M	Sensor S is malicious
T_E	Total number of examined sensors
T_M	Total number of malicious sensors
T_{M_F}	Total number of malicious sensors that fail a given test
TPR	True Positive Rate
FPR	False Positive Rate

3.2 ATTESTATION VERSUS MEASUREMENTS INSPECTION

Attestation and measurements inspection are very different in nature. Besides the difference in performance, they also vary significantly with respect to constraints introduced, test frequency, power overhead, and even threat model. We analyse these aspects below. Table 3 summarizes the notation used.

3.2.1 *Measurements Inspection Performance*

Measurements inspection is a multistep process consisting of detection, characterisation, and diagnosis; where the probability of succeeding in each step depends on the success of the previous steps. Success is determined by two factors: identifying as malicious only malicious scenarios (true positives) and not genuine scenarios (false positives).

Moreover, the performance of detection is different from the performance of characterisation and diagnosis. Indeed, detection operates at the granularity of the network, or of a cluster of nodes, while characterisation and diagnosis operate at the granularity of sensors. For this reason, we can link the performance of detection to the random variable C, representing the number of malicious nodes in the network. Instead, the performance of characterisation and diagnosis are linked to the random variable S_M, representing that a generic sensor S is malicious (or dually S_G, the event that a generic sensor S is genuine, where $P(S_G) = 1 - P(S_M)$).

Considering the whole measurements inspection process, a true positive is a malicious sensor that, after anomaly detection is triggered, is characterised as anomalous and diagnosed as malicious. A false positive, instead, is a genuine sensor diagnosed as malicious, after being characterised as anomalous. This can occur, e.g., because the anomaly is falsely detected or the characterisation falsely blames that sensor. Considering the three steps separately, the probability of having a true positive from measurements inspection is:

$$\begin{gathered} P(MI_F^S|S_M) = P(A_{D_T} A_{C_F^S} A_{D_M^S}|S_M) = \\ P(A_{D_M^S}|A_{C_F^S} A_{D_T} S_M) P(A_{C_F^S}|A_{D_T} S_M) P(A_{D_T}|C > 0) \end{gathered} \tag{1}$$

Where MI_F^S is the event that sensor S fails measurements inspection, A_{D_T} denotes the event that the anomaly detection algorithm triggers, $A_{C_F^S}$ the event that characterisation fails, and $A_{D_M^S}$ the events that the sensor is diagnosed as malicious.

Whereas, the probability of a false positive is:

$$\begin{gathered} P(MI_F^S|S_G) = P(A_{D_T} A_{C_F^S} A_{D_M^S}|S_G) = \\ P(A_{D_M^S}|A_{C_F^S} A_{D_T} S_G) P(A_{C_F^S}|A_{D_T} S_G) \\ \Big(P(C = 0) P(A_{D_T}|(C = 0) \cancel{S_G}) + P(C > 0) P(A_{D_T}|(C > 0) \cancel{S_G}) \Big) \end{gathered} \tag{2}$$

In the absence of a probabilistic model for both approaches, the probabilities can be approximated with experimental frequencies. In particular, the probabilities listed above can be characterised with True Positive Rates (TPR) and False Positive Rates (FPR). Indeed, the frequency of $(MI_F^S|S_M)$ and $(MI_F^S|S_G)$ correspond to the measurements inspection TPR and FPR, which we refer to as TPR$_{MI}$ and FPR$_{MI}$. We give a proof for the former, while an analogous proof holds for the latter. Let T_E denote the total number of examined sensors, T_M denote the total number of malicious sensors within T_E, and T_{M_F} denote the total number of malicious sensors that fail measurements inspection, then:

$$P(MI_F^S|S_M) = \frac{P(MI_F^S \cap S_M)}{P(S_M)} \approx \frac{\frac{T_{M_F}}{T_E}}{\frac{T_M}{T_E}} = \frac{T_{M_F}}{T_M} = TPR_{MI} \tag{3}$$

With a similar reasoning the $P(MI_F^S|S_G)$ can be approximated with the measurements inspection FPR (FPR$_{MI}$).

3.2.2 *Attestation Performance*

Attestation mechanisms are designed in such a way that compromised devices should not be able to compute a correct response, at least not within a given time limit. In theory, a malicious device S_M always has a chance ϵ of correctly replying in time, for instance, in case of a collision — when the genuine and modified memory contents output the same checksum value — or if the device's memory is traversed in a pseudo-random fashion and a modified memory address is not checked. Nonetheless, if the assumptions made by the attestation mechanism in place hold, then this chance should be negligible. Thus:

$$P(A_{T_F^S}|S_M) \geq 1 - \epsilon \tag{4}$$

Whereas, a genuine node should always pass attestation. Thus, the probability of a false positive is:

$$P(A_{T_F^S}|S_G) = 0 \tag{5}$$

We can characterise the attestation TPR, which we refer to as TPR$_{AT}$, by analysing the frequency of the event $(A_{T_F^S}|S_M)$. Let T_E denote the total number of examined sensors, T_M denote the total number of malicious sensors, and T_{M_F} denote the total number of malicious sensors that fail attestation, then:

$$P(A_{T_F^S}|S_M) = \frac{P(A_{T_F^S} \cap S_M)}{P(S_M)} \approx \frac{\frac{T_{M_F}}{T_E}}{\frac{T_M}{T_E}} = \frac{T_{M_F}}{T_M} = TPR_{A_T} \tag{6}$$

By definition FPR$_{A_T} = 0$.

3.2.3 *Comparison*

Constraints Introduced. Measurements inspection introduces the need for a trusted *inspector*, i.e., an entity which can run the anomaly detection algorithm and be trusted for its output. Moreover, good performance is achieved when data from many sensors is available. Therefore, it is often cheaper to make a centralized device in charge of running the algorithm, such as the base station or a tamper-proof local coordinator node. Similarly, attestation introduces the need for a trusted *verifier* to attest untrusted nodes. Furthermore, different assumptions are made according to the attestation mechanism in place. For instance, hardware-based and hybrid approaches require sensor nodes to be equipped with specific hardware, while software-based techniques nodes impose other restrictions, such as provers being in communication range of the verifier, and so forth.

Test Frequency. Measurements inspection ascertains the measurements integrity at a specific time. If the application layer aggregates multiple time samples, then it is convenient to run anomaly detection on such aggregates. This results in a better aimed protection of the application task and in a reduced detection frequency. Anomaly detection can be run on aggregates even when the measurements of each time instant are used separately, provided a mechanism that extends the validity of a detection check to future samples. Attestation has analogous requirements. Measurements produced during the time of a successful attestation are genuine. However, the sensor node may misbehave before or after the verification is complete. The larger the time gap between attestation and measurements transmission, the smaller the reliability.

Power Overhead. When anomaly detection is run by the sink, there is no communication overhead. On the contrary, such entity is subject to a computational overhead that is $\mathcal{O}(NlogN)$ in the number of measurements [110]. Attestation instead, introduces a communication overhead, due to the attestation protocol, and a computational overhead to the nodes, which need to compute a response, as well as the verifier, which needs to validate every prover's response. Both in computation and communication, the overhead of attestation is noticeably higher than measurements inspection.

Threat Model. On the one hand, attestation mechanisms can detect any types of attacks, e.g., targeting integrity, confidentiality or availability aspects, as long as the adversary modifies the software of compromised devices. On the other hand, measurements inspection can only expose attacks targeting the measurements integrity, but it can do so even in cases attestation would not work, for instance when the adversary physically tampers with environmental conditions, such as lighting a match close to a sensor.

3.2.4 *Trade-off Tuning*

Highly Reliable Attestation. Attestation is more reliable when the Attestation Frequency (AF) is high. Each individual sensor has its own attestation frequency, defined as the number of attestation challenges received divided by the number of measurements sent by the same sensor. In the following, we refer to AF as the attestation frequency averaged across all nodes. For instance, if only one sensor is attested each time it sends a measurement, then $AF = 1/N$. Assuming all sensors are equally important, they should all be attested with the same frequency. In such case, the reliability of attestation is high when $AF \approx 1$, i.e., when an attestation response is sent together with each measurement report. However, with such choice, the communication overhead is three times as high on average (because of the challenge and response messages). A noticeable computational overhead is also present for the calculation of the attestation response. In conclusion, using $AF \approx 1$ reduces the network lifetime, requiring a high maintenance cost either to replace nodes battery or to insert new nodes.

Low-Power Attestation. The cost of attestation can be reduced by decreasing AF. This can be done by either increasing the time between two attestations or reducing the number of attested sensors. However, the reliability of attestation deteriorates when the time between two attestations, i.e., $\frac{T}{AF}$, is close to the time needed to swap the genuine software with a malicious one, where T is the time between two measurement transmissions. A possible way to reduce the power of attestation, is then to reduce the number of attested sensor nodes. This can be done by selecting a random subset of nodes to attest which is small compared to the total N nodes. Similarly to the detection step in measurements inspection, when attestation fails for at least one sensor, a more thorough analysis can be triggered, i.e., attesting all other nodes. Such "attestation-based detection" is reliable only when the number of attested nodes is comparable to $N - C$, i.e., the number of genuine sensors. Therefore, unless almost all sensor nodes are compromised, the power overhead cannot be reduced significantly without a significant loss in reliability.

Low-Power Measurements Inspection. Measurements inspection makes a reliable measurements integrity check during the detection step: when the false information introduced by malicious sensors is far from reality and there are genuine sensors whose correlation with malicious sensors is disrupted, detection unveils the presence of malicious data. This is generally the case when there are at least G genuine sensors, where G depends on the WSN deployment, on the monitored physical phenomenon, and on the kind of malicious measurements. Measurements inspection is also able to identify malicious sensor nodes in the characterisation step. However, for this step to be as successful as detection, a higher value for G is required. To keep power consumption at a minimum, measurements inspection needs to accept the presence of potentially malicious sensors and identify only the most likely compromised. Thereafter, if the remaining malicious nodes keep injecting malicious data, the attack becomes less efficient, and they need either to make the false measurements closer to reality or to become more detectable.

Highly Reliable Measurements Inspection. To increase the number of discovered malicious nodes, measurements inspection needs to rely more on detection and less on characterisation. For instance, characterisation can produce just a set of mutually exclusive hypotheses for the detected anomaly with the correspondent malicious sensors. Further investigations, conducted in a reliable way, would then reveal which hypothesis is correct and which sensors are malicious. This can be done by checking sensor nodes in the field, but on the other hand it is against the goal of reducing the maintenance cost and measuring the WSN health from within itself.

In conclusion, attestation can ascertain the measurements' integrity, provided that it is run close to their transmission time. Running attestation for all measurements is expensive, but running it for an arbitrary subset decreases reliability. Measurements inspection can detect the presence of malicious measurements and identify suspicious sensors, but the final decision about a sensor's maliciousness should be taken with a more reliable approach. The two techniques complement each other as measurements inspection can trigger attestation in anomalous scenarios only, reducing the attestation frequency and keeping reliability high.

3.3 ASSUMPTIONS AND ADVERSARY MODEL

We assume an attacker whose goal is to evoke fake events or conceal real ones while remaining undetected. In particular, we consider the case where the adversary compromises sensor nodes modifying their software to perform malicious data injection attacks. Such malware can: (i) introduce a mismatch between the measurements observed by the sensor and the ones reported, which are referred to as *malicious measurements*, (ii) make the compromised sensors inject malicious measurements according to a common strategy, (iii) overhear the measurements sent by genuine nodes, and (iv) adapt malicious measurements in function of genuine ones to reduce the risk of introducing detectable anomalies.

While, individually, attestation could also detect other types of attacks, e.g., to confidentiality and availability, and measurements inspection could detect physical attacks tampering measurements, we concentrate on the collaboration between both mechanisms and leave a scenario with such events as future work.

We assume the adversary is in control of a subset of sensor nodes. The premise that some nodes are not compromised is a consequence of the cost of compromising them. In practice this usually holds given that sensors are not identical (e.g., different generations) even when they monitor the same phenomenon; some sensors may be physically hard to reach while others are easier; and some sensors will be subject to maintenance (e.g., because of fouling). A scenario where all nodes are compromised is, therefore, extreme.

Given that, due to low cost requirements, typical sensor nodes are not equipped with tamper-resistant hardware, we consider the use of software-based attestation. More specifically, we use SWATT [44], a software-based attestation mechanism with strict timing conditions. This introduces a series of other assumptions: (i) the base station, which plays the role of the verifier, cannot be compromised, (ii) all provers are within direct communication range and can only communicate with the verifier during attestation, (iii) there is an authentication system in place, (iv) the adversary cannot modify the nodes' hardware, (v) the implementation of the attestation routine is time-optimal, and (vi) the network maximum RTT is known a priori and is used to define the attestation timeout in such a way that genuine nodes are always able to correctly reply in time. Note that we can combine measurements inspection with any kind of attestation mechanism and, as long as nodes possess the specific hardware required by hardware-based or hybrid approaches, we can generalize the results of our study to such techniques while at the same time relaxing the assumptions necessary for the use of software-based attestation.

Lastly, we use the measurements inspection scheme described by Illiano et al. [110], which provides detection of sophisticated collusion attacks and also includes a characterisation and a diagnosis procedure.

3.4 COMBINATION STRATEGIES

The objective of the combination is to achieve a trade-off with much higher security confidence than measurements inspection, and a decrease in power overhead compared with attestation. However, their combination gives rise to a full spectrum of solutions in the trade-off between power efficiency and security, depending on how and when measurements inspection hands over to attestation. Predicting the resulting performance for a given choice is complex, as it depends on the interdependence of the techniques. Moreover, we wish to perform a preliminary study the potential of the combination independently from the performance of each single approach, to extract information that holds in general, and subsequently consider specific state-of-the-art techniques. For this reason, we present three different combination approaches and express their performance as a function of the variables involved.

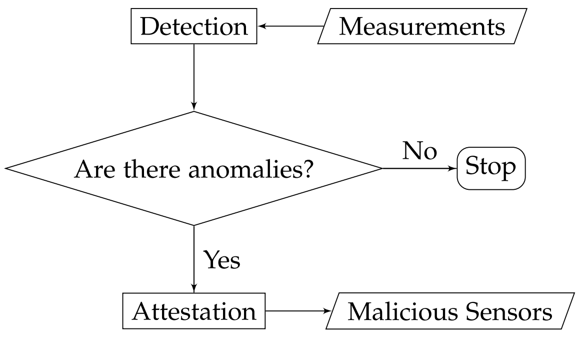


Figure 5: D&A Scheme.

3.4.1 *Detect and Attest*

The architecture of our first proposed combination, denoted with *Detect and Attest* (D&A), is shown in Figure 5. This scheme is designed to exploit only the most reliable step in measurements inspection, which is detection, and relay the characterisation task to attestation.

3.4.1.1 *Performance*

Since D&A is the series of the detection step from measurements inspection and the result of attestation, the output TPR is the product of the TPRs of both:

$$TPR_{D\&A} = TPR_{AD}TPR_{AT} \tag{7}$$

A false positive occurs when measurements inspection detects an anomaly, regardless of the presence of malicious data, and attestation fails on a genuine node. Thus, the FPR of D&A is:

$$FPR_{D\&A} = \Big(P(C=0)FPR_{AD} + P(C>0)TPR_{AD}\Big)FPR_{AT} \tag{8}$$

Where the term in brackets is the probability that anomaly detection triggers. This coincides with the expected attestation frequency, since all sensor nodes are attested when anomaly detection triggers:

$$AF_{D\&A} = P(C = 0)FPR_{AD} + P(C > 0)TPR_{AD} \tag{9}$$

3.4.2 *Group Subset Attestation*

In our second proposed combination, *Group Subset Attestation* (GSA), measurements inspection is used in the detection step and produces the groups of possible malicious nodes, while attestation acts as a judge to take the final decision. Its architecture is depicted in Figure 6. We observe that GSA uses measurements inspection for detection of anomalies and the grouping part of characterisation. Then it iteratively selects a random member for each group and attests it. When the ratio of genuine sensors in a group outweighs the ratio of malicious sensors or vice versa, the attestation for that group stops and the result for the majority of sensors is applied to the whole group. The guard condition that determines whether the number of attestations is high enough for the group is:

$$\frac{\| |S \in g : S_G| - |S \in g : S_M| \|}{|g|} > \delta_{GSA} \tag{10}$$

This condition makes sure that the ratio of genuine sensors in a group outweighs the ratio of malicious sensors by δ_{GSA} or vice versa. GSA keeps attesting all nodes in a group until the condition is met. When $\delta_{GSA} = 1$, the condition is never met and GSA coincides with $D\&A$, but with lower values, the number of attestation is lower compared to $D\&A$ and GSA may be more convenient.

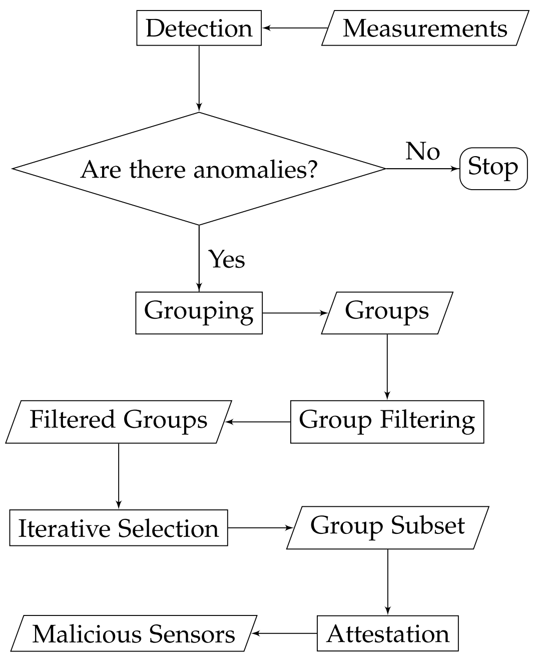


Figure 6: GSA Scheme.

To maximise energy efficiency, *GSA* may attest only one node per group. However, this choice is only accurate if the nodes in a group are all genuine or all malicious, which in general cannot be guaranteed. Indeed, measurements inspection is highly accurate in the grouping from the measurements perspective, i.e., the measurements in one group are either genuine or malicious. However, integrity violations at the node layer cannot be identified if the measurements are genuine. When used as a standalone technique, the identification of such malicious sensors is just delayed to the moment when the measurements become malicious. With GSA, if a node runs a malicious software but reports genuine data at the time it is attested, we would infer that all nodes in the same group are malicious. This, in turn, causes a wrong detection of non-compromised nodes whose measurements correlate with that of the malicious node.

The optimal value of δ_{GSA} makes an exhaustive attestation of groups that are mixtures of genuine and malicious nodes, and only one for groups where nodes are either all genuine or all malicious. The value used in our simulations (Section 3.6) is 0.25, which gives a percentage of attested nodes around 25%.

3.4.2.1 *Performance*

Note that for each group only a subset of sensors is tested, hence the TPR is:

$$\begin{aligned} TPR_{GSA} = \delta_{GSA} TPR_{AD} TPR_{AT} + \\ (1-\delta_{GSA}) \Big(P(S'_M | g(S) = g(S') S_M) P(A^{S'}_{T_F} | S'_M) + \\ P(S'_G | g(S) = g(S') S_M) P(A^{S'}_{T_F} | S'_G) \Big) TPR_{AD} \end{aligned} \tag{11}$$

Where $P(A^{S'}_{T_F} | S'_M) \approx TPR_{AT}$ and $P(A^{S'}_{T_F} | S'_G) \approx FPR_{AT}$.

We denoted with TPR$_{AD}$ the TPR of the anomaly detection part of measurements inspection, and with $P(S'_M | g(S) = g(S') S_M)$ the probability that the sensor chosen for attestation S' is malicious given that the considered sensor S is malicious and belongs to the same group as S'. Namely, if the malicious sensor node is an attested node (they are δ_{GSA} of the total on average), then detection is correct if anomaly detection triggers and the node fails attestation. Otherwise, the characterisation is correct only if the correct group is selected. Thus, the FPR becomes:

$$\begin{aligned} FPR_{GSA} = \Big(P(C=0) FPR_{AD} + P(C>0) TPR_{AD} \Big) \\ \Big(FPR_{AT} \delta_{GSA} + (1-\delta_{GSA}) \Big(P(S'_M | g(S) = g(S') S_G) TPR_{AT} \\ + P(S'_G | g(S) = g(S') S_G) FPR_{AT} \Big) \Big) \end{aligned} \tag{12}$$

The attestation frequency here is equal to:

$$AF_{GSA} = \Big(P(C=0) FPR_{AD} + P(C>0) TPR_{AD} \Big) \delta_{GSA} \tag{13}$$

Compared with D&A, the TPR and FPR are as good or better if the anomaly-based grouping is correct and worse otherwise, while the attestation frequency is always lower except for the cases where δ_{GSA} equals to 1 or if all groups consist of single nodes.

3.4.3 *Cascade*

If the strict time constraints in attestation are always guaranteed by genuine sensor nodes, a failed challenge response is an undisputed proof of a sensor's maliciousness. A malicious node found with anomaly detection, instead, may be the consequence of an unforeseen or unprecedented scenario that caused a wrong estimate of the measurements probability. Hence, attestation can be used to confirm a sensor node's maliciousness after it has been identified as such by the measurements inspection, as shown in Figure 7. We refer to this approach as *Cascade*.

3.4.3.1 *Performance*

The TPR and FPR of the scheme are:

$$TPR_{Cascade} = TPR_{MI} TPR_{AT} \tag{14}$$

$$FPR_{Cascade} = FPR_{MI} FPR_{AT} \tag{15}$$

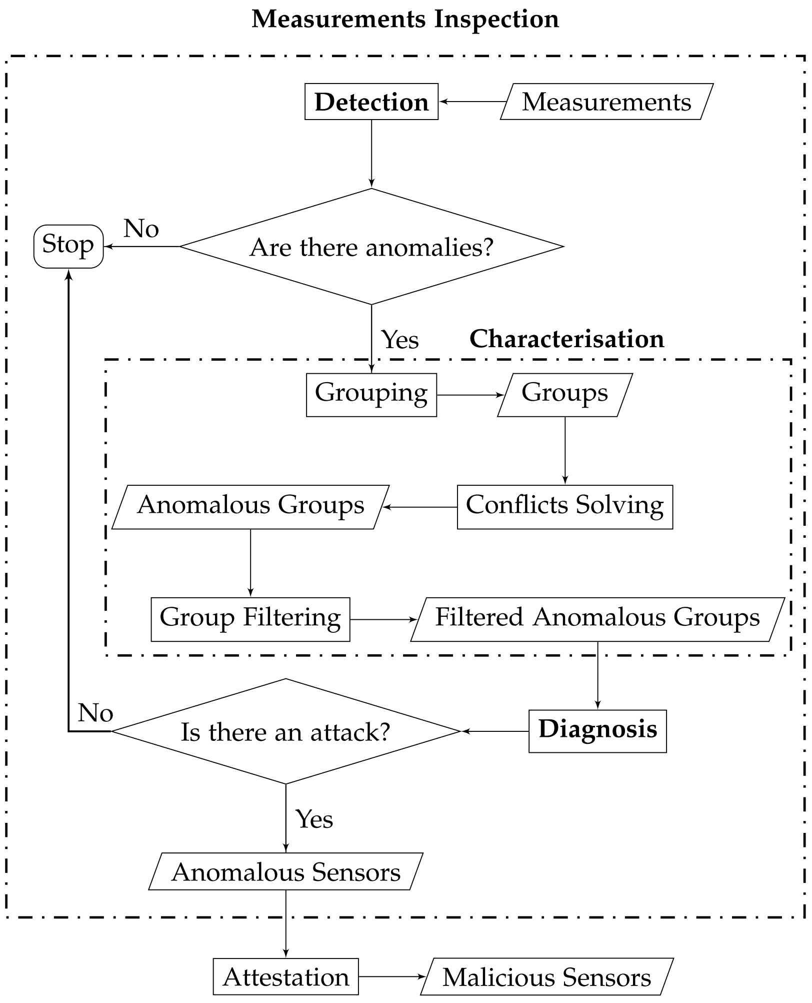


Figure 7: Cascade Scheme.

The Cascade scheme power overhead is lower than attestation since the attestation frequency is:

$$AF_{Cascade} = P(C > 0)TPR_{MI} + P(C = 0)FPR_{MI} \tag{16}$$

This is also lower than the attestation frequency of GSA, when the measurements inspection FPR and the number of malicious nodes are low. Instead, with higher FPR, the

Cascade scheme makes many attestations, while the number of attestations for the GSA scheme is upper bounded by the number of groups given in output by characterisation.

Note that if the attestation mechanism assumptions hold then $\text{TPR}_{AT} = 1$ and $\text{FPR}_{AT} = 0$, so, compared with attestation, the Cascade solution trades a reduced power overhead for a lower TPR, which coincides with that of measurements inspection.

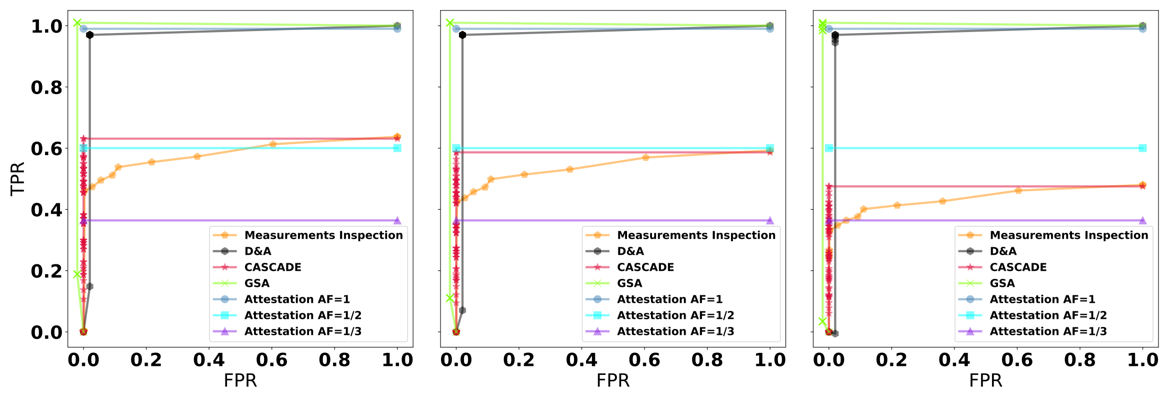
3.5 ANALYTICAL EVALUATION

In this section we evaluate the techniques in a WSN of 200 sensor nodes, of which a varying number C are compromised. Sensors collect a measurement about every 4 minutes. The probability of attack is as high as 10^{-2}, implying that, on average, there is an attack about every 7 hours. This value is in the same order of FPR_{AD}, so the analytical results remain substantially unchanged also for lower attack probabilities. When an attack occurs, the probability that a sensor is malicious has a uniform prior distribution across all sensors (i.e., each sensor is malicious with probability $1/C$).

The evaluation is done by calculating: (i) The *Receiver Operating Characteristic* (ROC) curves, i.e., the relationship between TPR and FPR in the identification of malicious nodes. (ii) The attestation frequency, i.e., the time between two attestations divided by measurements transmission period, and averaged across all nodes.

The ROC curves of the combination schemes are obtained through the results of Section 3.4. The ROC curves for measurements inspection are obtained by interpolating the experimental curves in [110]. The ROC curves for attestation are obtained by modelling the degradation in the reliability of attestation, when the time between two attestations is close to the time needed to swap the genuine software with a malicious one, and vice versa. Thus, the probability that an attacker manages to compromise a measurement and pass attestation behaves like an exponential probability distribution:

$$1 - e^{-\lambda(\frac{1}{\text{AF}}-1)} \tag{17}$$



(a) C = 50 (b) C = 100 (c) C = 150

Figure 8: ROC curves comparison varying the number of malicious sensors C[1].

The parameter λ models the time needed to replace the genuine software and restore it. In particular, we show the performance of attestation when $\lambda = 0.5$, which is the case where the probability that a malicious node passes attestation is close to 0 when $AF = 1$, and quickly increases to about 0.4 and 0.6 for $AF = 1/2$ and $AF = 1/3$, respectively.

3.5.1 *ROC Curves Comparison*

Figure 8 shows the ROC curves for both individual and combined techniques. In particular, the TPR of attestation is shown for $AF = \{1, 1/2, 1/3\}$, corresponding to one attestation run as soon as a new measurement is collected, or once every two or three measurements are collected. The TPR decreases with the attestation frequency since a quick attacker may substitute the original software with a malicious one, inject malicious data, and replace the original software before attestation is run again.

Comparing Figures 8 (a), 8 (b), and 8 (c), the measurements inspection TPR curves appear to saturate at a value around 0.5 that decreases as C increases. This is an effect of the conservative group-wise characterisation [110] which, on the other hand, is the tool that

1 A perturbation of ± 0.02 was introduced in GSA and D&A to better distinguish the curves.

keeps the FPR close to 0 where the TPR is around 0.4. The highest TPR of measurements inspection, achieved at the rightmost point of the ROC curve, is an upper bound to the TPR of Cascade. Nevertheless, while measurements inspection achieves such TPR with FPR close to 1, Cascade achieves it with FPR close to 0. The performance of D&A and GSA is not limited by measurements inspection since they mainly exploit the detection step, whose performance is comparable to attestation's. Indeed, their ROC curves nearly overlap with that of attestation.

3.5.2 *Attestation Frequency Comparison*

To make a fair comparison, we also consider the attestation frequency of each scheme, since with $AF \approx 1$ there is no advantage in using a combination scheme in place of simple attestation.

Figure 9 shows that the points where Cascade performs at its best are expensive, as the attestation frequency is close to 1. Instead, when the attestation frequency of D&A is as low as 0.02, the TPR in Figure 8 is close to attestation. Finally, GSA has a small attestation frequency which saturates and holds even for the highest values of TPR. In conclusion, Cascade is generally not convenient, since it covers a point in the reliability-cost space where reliability is close to measurements inspection and cost is close to attestation. For D&A and GSA, the reliability is almost as high as attestation. The cost for GSA is always comparable to measurements inspection. For D&A, the cost can be kept low for a low decrease in reliability.

3.6 NUMERICAL SIMULATIONS

In the previous section, we abstracted from the computations and network protocols that enable the application of each combination scheme, so we address them below.

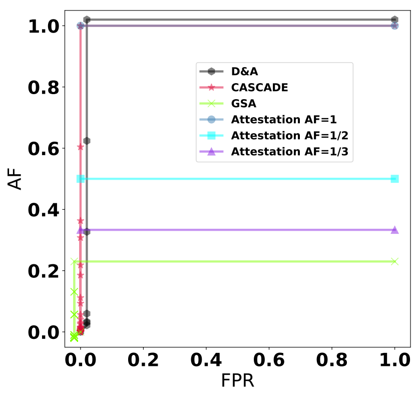


Figure 9: AF curves comparison[1].

Preliminarily, we address the process of tailoring attestation for measurements reliability, which has not been analysed in the literature yet.

Moreover, we validate the analytical ROC curves and complete them with the time needed to achieve a certain TPR, which corresponds to the latency in the reaction to malicious data. Finally, we make an accurate estimation of the energy spent, which can be obtained only after network protocols are defined. This allows us to calculate the impact of each approach on nodes' battery life.

To investigate accurately these parameters, we have set up simulations of a realistic application scenario in the open-source *Castalia* [111] simulator. We recur to simulations as existing datasets do not contain sophisticated injection attacks such as those we consider. Moreover, simulations allow running both the individual and combined schemes under exactly the same scenario, giving accurate comparisons that highlight the gains in performance and energy consumption, and allow evaluation with more devices than it would be practical with real nodes.

3.6.1 *Simulation Settings*

The simulations consider 200 nodes in a star-topology network, where the base station is located at the centre performing the role of network coordinator, measurements sink, and attestation verifier. Sensors measure the temperature and send the observed value to the sink about every 4 minutes. Nodes are equipped with the CC2420 RF transceiver [112], which is common for its low-power transmissions. Namely, this device spends 57.42 mW while transmitting (at 0dBm), 62 mW while receiving and 1.4 mW while in sleep mode [112].

For C out of 200 sensor nodes, the temperatures sent to the sink are replaced with higher values to trigger the detection of a wildfire. This is done with the data injection technique described in [110], which aims to overcome anomaly detection by minimising the maximum expected correlation between genuine and compromised sensors. Indeed, if a malicious measurement is expected to be highly correlated with a genuine measurement, significant changes in the former would introduce obvious anomalies. The simulations are run for all individual and combination schemes, with three different values of C: 50, 100, and 150. After 5 hours the simulation is stopped and we calculate the TPR and FPR for the detection of malicious nodes, and the energy consumption.

We use the 802.15.4 MAC protocol [113] because it provides us low energy consumption and a hierarchical architecture. The protocol makes use of a coordinator node that dictates how and when nodes can communicate through the use of a superframe structure, depicted in Figure 10, which constitutes an active and an inactive period, where nodes are allowed to transmit or switch off their communication devices to save energy, respectively. Furthermore, at the application level we define a transmission schedule so that nodes transmit their measurements one after the other, thus avoiding collisions. Once the sink receives all measurements it can trigger the measurements inspection, attestation, or a combined scheme. Note that when just measurements inspection or no scheme is in use, the nodes can go to sleep right after they transmit their data. Whereas, if a scheme with attestation is in place, the nodes must keep awake as they do not know in advance whether they will be attested. Since the communication pattern is not fixed we cannot use the canonical 802.15.4 guaranteed time slots. Also to avoid collisions, so that the network maximum round-trip time can be reliably estimated, attestation is performed one node at

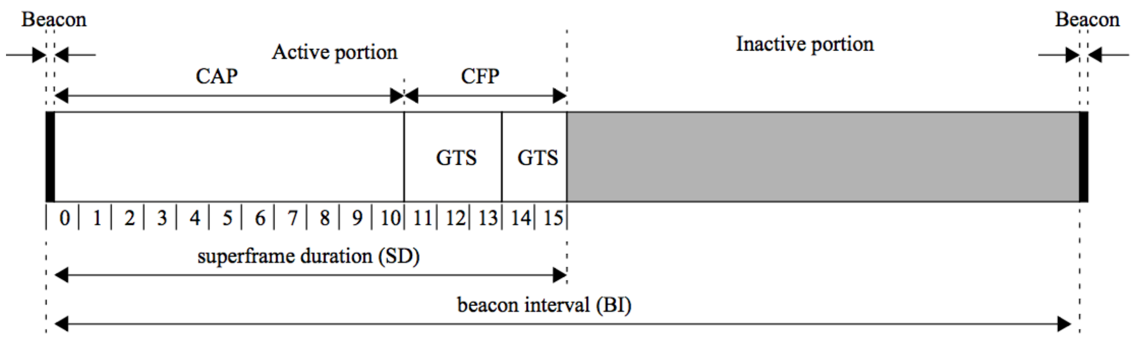


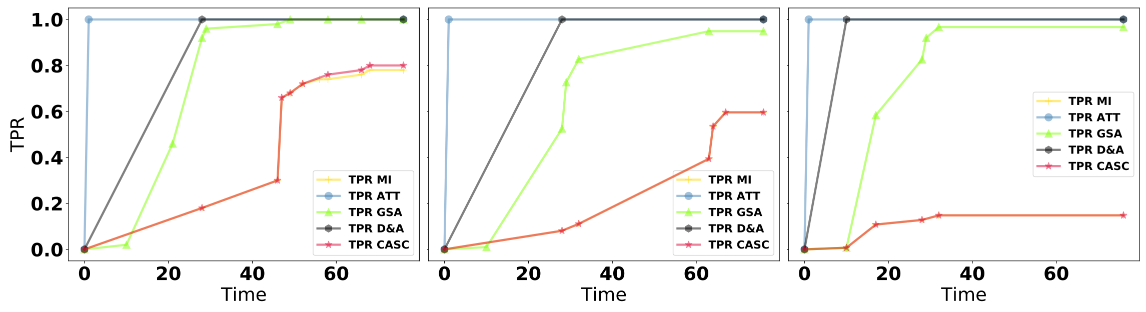
Figure 10: 802.15.4 MAC superframe.

a time. When the sink is done attesting all nodes for a collection round, it sends a message signalling to nodes that they can go immediately to sleep until the next active period.

3.6.2 *True Positives / False Positives Results*

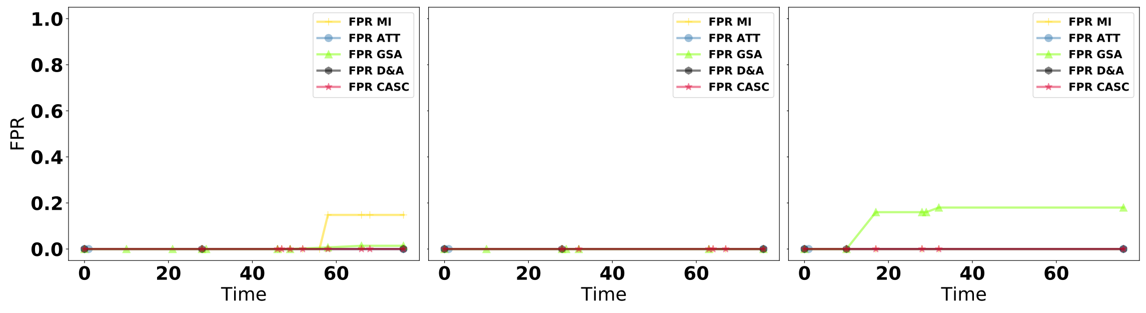
We calculate the TPR, defined as the number of malicious nodes that are detected at least once during the simulated attack. We do not assume the presence of *intrusion reaction systems*, therefore a detected node keeps carrying out the attack. This constitutes an upper bound on the system's performance since both the energy consumption and the detection of further nodes improves with the correct characterisation of malicious nodes. Indeed, a malicious node that is detected does not need to be evaluated again by either attestation or measurements inspection. The performance of the latter is also likely to increase, since less malicious nodes have less benefits from collusion.

Figure 11 shows the TPR and FPR for each scheme with different numbers of malicious sensors. Rather than their final values, we show the cumulative TPR and FPR time series. Indeed, the time needed to achieve a certain TPR is decisive to minimise the attack's damage. Analogous considerations hold for the FPR. The combination technique with the best performance is certainly D&A, whose curves are the closest to those of attestation. Since D&A uses the detection step of measurements inspection, the TPR curves, in Figures 11 (a),



(a) $C = 50$ (b) $C = 100$ (c) $C = 150$

Cumulative TPR in time.



(d) $C = 50$ (e) $C = 100$ (f) $C = 150$

Cumulative FPR in time.

Figure 11: TPR/FPR curves comparison with different numbers of malicious sensors C.

11 (b), and 11 (c), jump from 0 to 0.99 after 28, 28, and 10 collection rounds respectively, which are the points where anomalies are detected and attestation is triggered.

The curves of the Cascade scheme are close to measurements inspection, but an improvement is brought in the FPR curve in Figure 11 (d), which is decreased from 0.15 to 0. Higher TPR can be achieved by increasing the measurements inspection FPR and letting attestation take care of the false positives by attesting them. However, this choice increases the attestation frequency, as discussed in Section 3.5. The simulations also confirmed that D&A can achieve an FPR close to 0 and a TPR around 0.99. In the case of GSA the TPR is around 0.96 and the FPR is close to 0, especially when 50 and 100 sensor nodes out of 200 are malicious. When 150 sensors are malicious the FPR is evidently higher than D&A, but the TPR that eventually gets close to 1 makes GSA still a valid choice. Indeed, when the system detects that 150 out 200 sensor nodes are malicious, it means that a severe attack

Table 4: Average energy consumption (Joules) after 5 hours.

C	Scheme					
	NONE	MI	ATT	GSA	D&A	CASC
50	79.54	82.86	109.84	83.65	91.14	83.59
100	81.51	80.96	116.65	80.91	83.73	80.04
150	82.01	79.67	125.81	80.45	83.35	80.07

has taken place. In such scenario, generating false positives is not the main concern since the system needs a thorough recovery and reconfiguration process anyway.

3.6.3 *Energy Consumption Comparison*

In Table 4, the energy consumption during simulation time, averaged across all nodes, is reported for each scheme. First, we note that attestation has an energy consumption which is between 33% and 58% higher compared with measurements inspection, and that the latter does not introduce a perceptible increase with respect to the case where no security scheme is applied (NONE in Table 4).

Compared with measurements inspection, GSA and Cascade generally demand less than 1% extra energy . The increase for D&A, instead, is between 3 and 10%. To understand practical implications of such differences in energy consumption, we used the results in Table 4 to retrieve the expected number of days until the batteries would drain. As a reference, we assumed a typical power source of two alkaline long-life AA batteries, which store an energy of 18720 Joules [114]. As reported in Table 5, we see that the average duration of the batteries without measurements inspection nor attestation is about 48 days. When running measurements inspection, there is no significant change in battery life. With attestation instead, the batteries last 10 to 15 fewer days. GSA and Cascade generally cause the batteries to last 1 fewer day, while with D&A battery life diminishes by 2 to 4 days.

Table 5: Days to battery depletion.

C	Scheme					
	NONE	MI	ATT	GSA	D&A	CASC
50	49.03	47.06	35.51	46.62	42.79	46.66
100	47.85	48.17	33.43	48.20	46.58	48.73
150	47.55	48.95	31.00	48.48	46.79	48.71

3.7 RELATED WORK

As we have seen in Chapter 2, the vast majority of works in the literature regarding attestation focus on the security of the integrity verification process. How often should attestation be performed, which nodes should be attested, and the impacts of performing it, especially in terms of energy consumption, are not well covered [14].

Chen *et al.* [93] investigate how often attestation should be triggered in order to optimize the network lifetime without degrading the detection rate of compromised nodes. They conclude higher compromise rates require higher attestation frequencies. However, it is difficult to know how fast an adversary can compromise network nodes. When attestation is used to ascertain the integrity of the measurements, the attestation frequency should be even higher, since there is a time lapse between the time a sensor node is attested and the time when the measurement is taken, in which the sensor node could potentially be compromised. This problem has not been addressed yet, and we solve it through the combination with measurements inspection, which decreases the attestation frequency by two orders of magnitude, regardless of the compromise rate.

With the ever increasing number of devices permeating our daily lives, scalability also becomes an issue. Asokan *et al.* [20], Ambrosin *et al.* [115], and Carpent *et al.* [116] examine how to scale attestation to efficiently verify a large number of devices. Our combination scheme allows us to target the problem from a different perspective. By requiring only a subset of devices to be attested we essentially circumvent the need for scalability, thus incurring less overhead on nodes, while still achieving a good detection rate.

The literature about measurements inspection is broad, but usually considers malicious interference analogous to genuine faults [109]. Instead, the adversary may seek to stay undetected. This is analysed only in a few works [117, 118, 119, 110]. The overhead in computations and communications introduced by such techniques is always kept low [109]. However, measurements inspection techniques have significantly poorer detection performance when several malicious sensor nodes collude in the injection of malicious data. For this reason, Tanachaiwiwat and Helmy [117] propose to deploy tamper-resistant sensor nodes that authenticate suspicious sensors. On the other hand, we are concerned with integrity problems rather than authentication and focus on detecting malicious activity in highly compromised networks even when nodes are not equipped with tamper-resistant hardware.

To the best of our knowledge, no other work in the literature presents a direct comparison between attestation and measurements inspection in WSNs, let alone their combination. We analyse and compare both approaches in detail, focusing on the aspects that make them complementary.

3.8 CONCLUSIONS

In this chapter, we show that combining attestation with measurements inspection achieves high accuracy in identifying malicious nodes whilst significantly reducing power consumption. We proposed three combination schemes: *Detect and Attest*, *Group Subset Attestation*, and *Cascade*. The first gives most relevance to the attestation step, the third stresses the measurements inspection steps, while the second is at a point in between. In this way, the spectrum of combinations is well covered.

We evaluated all schemes both analytically and under simulations. While the Cascade scheme has shown to be limited by the measurements inspection's maximum TPR, both D&A and GSA achieve a detection rate close to attestation, around 99% and 96% respectively, while keeping a power overhead close to measurements inspection, 10% and 1%.

This is confirmed by the energy results from the simulations and is due to the dramatic reduction in the number of attestations, which is observable in the analytical evaluation. As a result, a good trade-off between energy and performance is achieved by both D&A and GSA schemes. Whereas the individual techniques forced a decision between accuracy close to 100% with power overhead of 33-58%, or accuracy close to 50% with power overhead close to 0.

4 STOCHASTIC SOFTWARE-BASED ATTESTATION

In Chapter 3 we described how to combine attestation with measurements inspection and the benefits of doing so. More specifically, we imagined a scenario using software-based attestation with strict timing conditions and took precaution to respect its underlying assumptions. In particular, we took special care to avoid any collisions by having only one node transmitting at a time. Thus, we always respected the assumed maximum network round-trip time (RTT) used by the attestation mechanism to set its timeout, which is crucial to differentiate honest from compromised devices. However, in many cases it is not possible to guarantee a maximum network RTT. For instance, several applications do not have a transmission schedule and merely use a Medium Access Control (MAC) protocol to *avoid* collisions, often times these networks are deployed in an environment where the wireless medium is exposed to external interference, and nodes equipped with cheaper radio devices are more susceptible to unpredictable faults. Furthermore, the higher the attestation timeout, the higher the number of computations the untrusted device has to perform, consuming extra time and energy and restraining it from performing its primary tasks. Hence, using the maximum known network RTT to set the attestation timeout incurs a high overhead and still does not guarantee that an honest device will always be able to reply in time, which makes software-based attestation impractical in many scenarios.

In this chapter, we review this critical, and yet overlooked, assumption and propose a novel stochastic software-based attestation approach. Our idea is to break the standard single challenge-response protocol used by previous schemes into a sequence of challenges with shorter timeouts, requiring only one of them to be correctly answered within time. We take advantage of the fact that most of the time the actual network RTT is much smaller than the maximum known RTT. Thus, we can configure a sequence of challenges in such a way that there is a high probability at least one will be replied in time. While this approach significantly improves the attestation performance, it remains secure against all known attacks.

Given that we had no access to actual WSNs deployments but wanted to experiment our proposal under realistic conditions we implemented and evaluated it using IoT devices, the Intel Edison [120] platform, communicating over real-world uncontrolled Wi-Fi networks. To the best of our knowledge, this is the first time software-based attestation has been evaluated outside simulation or well-controlled environments.

The remainder of this chapter is organized as follows. We first give an overview of the design and implementation of a software-based attestation routine and how an attacker might attempt to bypass it in Section 4.1. Then, in Section 4.2, we review the maximum network RTT assumption, how it is used to set the attestation timeout, and why it is not practical. Afterwards, we describe the design of a new stochastic approach and assess its security in Section 4.3. We then provide an analytical, Section 4.4, and experimental, Section 4.5, evaluation comparing our proposal to the current state of the art solution. In Section 4.6 we examine other works in the literature that also deal with the network RTT. We draw our conclusions and discuss future work in Section 4.7.

4.1 SOFTWARE-BASED ATTESTATION ROUTINE

To put it simply a software-based attestation routine is essentially a self-checksumming code that takes an input from and reports its result to a verifier. However, its implementation has to respect a series of properties such that an adversary trying to bypass it will either compute the wrong result or take a distinguishable longer time to calculate the right one. In this work, we based our implementation on the routine proposed by Seshadri et al. [37] and we describe its main characteristics in this section.

4.1.1 *Implementation Required Properties*

Time-optimal. While this property cannot be formally guaranteed, the implementation makes use of simple instructions such as `add` and `xor` which are hard to compute with fewer or faster operations. In addition, the implementation is organized in code blocks in such a way that operations in one block depend on the results of the operations in the previous one, thus preventing reordering optimizations over blocks.

CPU state inputs. Incorporating the values of the data pointer as well as the program counter and status registers into the checksum serves to demonstrate that the code was executed as it was meant to be. Moreover, an adversary that changes any of these values will have to execute extra instructions to mask these changes and will suffer an execution time overhead.

Pseudo-random memory traversal. Reading memory addresses in a pseudo-random order prevents an attacker from knowing when a modified address is going to be accessed and forces it monitor all memory reads.

Iterative checksum code. The attestation routine contains a checksum computation loop that must be executed several times. Since the memory reads are inside the loop, to hide any modified memory content an adversary would have to introduce new instructions inside the loop, introducing a fixed overhead per iteration.

Strongly-ordered checksum function. The use of strongly-ordered functions prevents an adversary from reordering or parallelizing operations as there is a high probability the output will differ in such cases. For instance, the function $a_1 \oplus a_2 + a_3 \oplus a_4$ must be evaluated as $(((a_1 \oplus a_2) + a_3) \oplus a_4)$.

Keyed checksum. The challenge sent by the verifier is used to initialize the checksum to a deterministic but unpredictable state.

Small code size. A small code can fit on the processor cache achieving faster and more deterministic execution time. In addition, the adversary overhead is relatively higher the smaller is the checksum loop.

Instruction sequencing to eliminate empty slots. Modern CPUs are capable of issuing multiple instructions every clock cycle. Therefore, the checksum code instructions must be arranged in a sequence to avoid empty slots preventing an adversary from executing additional instructions with no overhead.

Low variance of execution time. Once it starts the attestation routine cannot be interrupted, thus it must execute at the highest CPU privilege level with all maskable interrupts disabled. Furthermore, as exceptions and non-maskable interrupts cannot be disabled, the attestation routine must replace their existing handlers with a stub handler that only executes an *interrupt return* instruction. Eliminating empty slots and having small code size also help to achieve a lower execution time variance.

4.1.2 *Implementation and Attack*

We have implemented our attestation routine inside the Edison's Wi-Fi kernel driver. This allows a node to disable interrupts and start computing the response immediately after receiving a challenge.

The main loop of the attestation routine consists of four code blocks as illustrated in Figure 12. The last instruction of each block is a jump to an absolute address that can point to the first or second instruction of any block. This address is also computed within the blocks using the current checksum value. Each block is 128 bytes in size and, to simplify

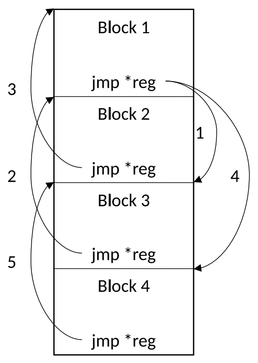


Figure 12: Checksum main loop structure [37].

the generation of the jump addresses, all blocks are aligned in memory in such a way that their first instructions are at addresses multiple of 128. Since the checksum function is strongly ordered, the result depends on the execution of the code blocks in the correct order, which in turn is defined by the sequence of jumps between blocks.

Seshadri et al. [37] describe a memory copy attack where the adversary runs a modified version of the attestation routine on the correct memory location and computes the checksum over a copy of the original version located somewhere else in memory. To do so it adds a constant displacement to the data pointer, which can be done in two ways: (i) annotating instructions that use the data pointer with a segment register, or (ii) using an addressing mode that adds an immediate to the data pointer. Both ways incur no execution overhead. However, they both increase the instruction length. To place the longer instructions, the adversary would then move the blocks farther away from each other aligning them to addresses multiple of 256, as shown in Figure 13. It can then generate the jump addresses by left-shifting the correct address by 1 before the actual jump. Nevertheless, the jump address is incorporated into the checksum both before and after the jump. Therefore, the adversary has also to restore the correct address by right-shifting it after the jump. Its overhead would then be of two instructions per iteration: one left-shift and one right-shift.

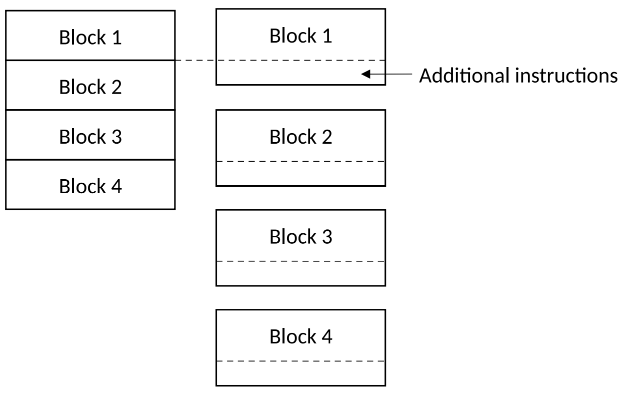


Figure 13: Attack structure [37].

It was only when we tried to implement this attack that we noticed it might not work as described. Note that when changing the block sizes the new addresses are not double the value of the original ones. Therefore, instead of shifting the whole value we only need to shift a few bits of the address. However, in the x86 architecture we can only shift all 32 bits of a register, its lower 16 bits, or 8 bits (0 to 7 or 8 to 15) at a time. Depending on the position of the original code, just using a single shift instruction might not result in the correct address, thus requiring additional instructions. While this is not a problem to generate the address before the jump, because we can use instructions from the original code to mask the result, it is a problem after the jump because at the beginning of a block there are no instructions computing the jump address.

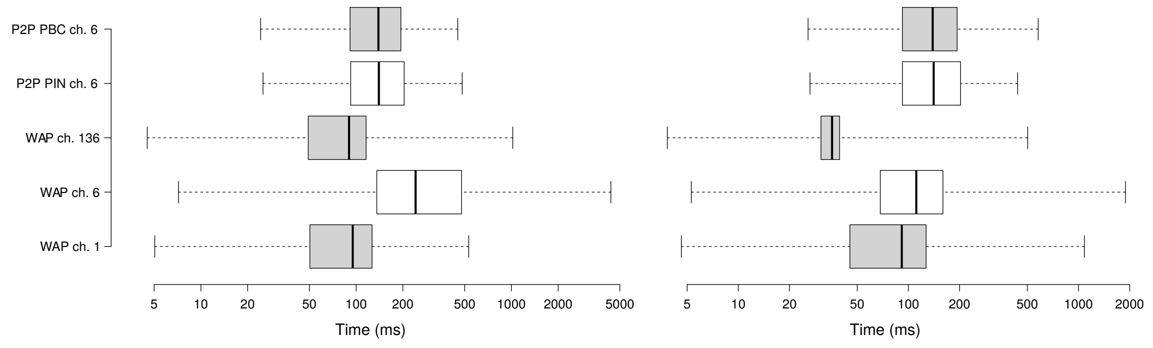
Gardner et al. [121] also base their implementation on the work of Seshadri et al. [37] and describe a variation of the same memory copy attack. They execute the malicious attestation code at its genuine location and place a copy of the original code one megabyte above it. To compute the checksum over the memory of the original code they `xor` the checksum target address by 0x100000 (one megabyte) immediately before the memory read instruction. After the read, they `xor` it again to recover its value before the address is incorporated into the checksum. The overhead of this attack is then two `xor` instructions per iteration. We note however, that they are only able to add these two instructions in place because their code blocks are not exactly 128 bytes long and do not prevent simple insertion of code.

In the end, we implemented an attack that combines ideas from the two attacks we just described. As done by Seshadri et al., we align the blocks of our malicious version at addresses multiple of 256. Before the jump we double the original address with an `add` instruction (this is similar to a left-shift by one, but has a smaller opcode) to get the modified address. After the jump each block code `xor`s the address with a different mask, thus recovering what would be the original address of the block. Therefore, our attack can successfully compute the right response with an overhead of only two instructions per iteration: one `add` and one `xor`.

4.2 THE MAXIMUM RTT PROBLEM

Existing software-based attestation mechanisms with strict timing conditions assume that by using the maximum known network RTT to specify the challenge timeout guarantees all honest devices can pass attestation. To show that this assumption is not practical, we went to the Imperial College Central Library to collect real data.

First, we scanned for available Wireless Access Points (WAPs) and Ad-hoc connections in range and found a total of thirty six connections. Many of them were using the same frequency, which leads to interference and performance degradation. From these, we had access to three different channels: 1, 6, and 136. We used two Intel Edison devices to measure the RTT. The Edison also supports Wi-Fi Peer to Peer (P2P) allowing devices to connect directly without the need for a WAP. It provides two modes of connection, Push Button Control (PBC) and PIN mode (PIN), which we also evaluated. Additionally, we measured the RTT with the Edison Wi-Fi driver power management, which is used to reduce power consumption, both on and off. Figure 14 shows the results obtained by using one device to ping the other a thousand times for each channel.



(a) Power Management On (b) Power Management Off

Figure 14: Box plot of the network round-trip time under different channels.

The variable transmission delays experienced are likely due to packet collisions, re-transmissions, and the use of CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) supplemented by RTS/CTS (Request to Send/Clear to Send) frames[1], all of which are influenced by the total number of devices concurrently using the network, the traffic generated by them, as well as external interference. Note that we had no control over these factors since we only owned two devices in the network and the measurements were not done all at the same time. With exception to channel 1, all channels present a smaller mean and standard deviation for measurements done with the power management turned off. Given all variables in play, it is difficult to give an explanation for the odd behaviour of channel 1, which is already an indicator that assuming a maximum RTT is unpractical and inefficient. To make the network RTT more deterministic, schemes such as SCUBA [42] assume that the prover has exclusive access to the radio channel during attestation. However, due to external interferences, the actual RTT may be worse than anticipated resulting in honest provers failing attestation. Nevertheless, for now, let us work with this assumption to estimate the necessary amount of attestation routine iterations.

1 While the use of RTS/CTS frames partially avoids the hidden terminal problem it also introduces the exposed terminal problem.

Since the attestation routine is assumed to be implemented in a time-optimal way, any modification an adversary makes to it will incur in computation time overhead. To compensate this overhead, the adversary may try to reduce the network RTT between the compromised device and the verifier in order to still reply within the attestation timeout. In theory, an adversary can have zero RTT, in which case its time advantage would be the maximum RTT allowed by the attestation process. Seshadri et al. also consider the time an adversary gets from warming up the cache to avoid cache misses during execution [37]. Nevertheless, when compared to the measured RTT samples in Figure 14, this gain is negligible. Because even the best adversary has an overhead per iteration, it is possible to compensate its time advantage, which is fixed, by increasing the number of iterations required.

To compute the number of iterations, three factors must be taken into account: c, the prover's CPU clock speed; a, the adversary time advantage; and o, the adversary overhead per iteration of the attestation routine [37]. Furthermore, because the memory is being traversed in a pseudo-random fashion, to guarantee, with high probability, that each memory address is accessed at least once, based on the Coupon Collector's Problem, the number of iterations for a memory of size m has to be at least $O(m \ln m)$ [78]. Hence the number of iterations i can be computed as below:

$$i \geq \frac{(c \cdot a)}{o} \geq m \cdot \ln(m) \tag{18}$$

The prover's CPU clock speed and the assumed adversary overhead per iteration are constants. Consequently the number of iterations becomes a linear function of the allowed maximum RTT as shown in Figure 15. The Edison CPU operates at 500 MHz, and our implementation of the best known attack, described in the previous section, has an overhead of 1.037 CPU cycles per iteration. The scenario that requires the smaller number of iterations, P2P PIN ch. 6 from the measurements in Figure 14 (b), would still need to execute 211463783 iterations. Based on the measurements we have done of the time

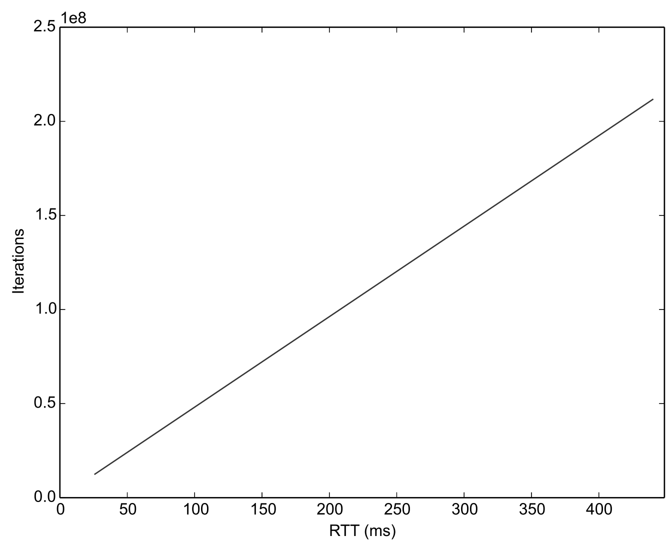


Figure 15: Number of iterations based on the RTT.

taken by each iteration of our implementation of the attestation routine it would take approximately 24.6 seconds to execute this number of iterations.

According to existing approaches, regardless of the actual network conditions, the assumed maximum RTT is always used to compute the number of iterations of the attestation routine. Therefore, even if the network conditions are better than estimated, the prover still has to compute a large number of iterations. For instance, as can be seen in Figure 14 (b), 75% of the RTT samples taken from P2P PIN ch. 6 take less than 202 ms and 50% of the samples are within the range of 26 to 139 ms. Nevertheless, a traditional attestation mechanism would always consider the maximum RTT of 438.682 ms to compute the number of iterations, wasting both time and energy. Furthermore, the higher the number of iterations, the higher the execution time jitter due to clock, cache, and TLB fluctuations. As observed by Li et al. [38], to avoid having honest provers not replying in time, the challenge timeout should also include the execution time jitter. However, an adversary could take advantage of this time extension to execute additional instructions and still pass attestation with non-negligible probability. Thus, a high execution time jitter forces a tradeoff between these two cases, both of which are not desirable.

The single challenge-response protocol followed by traditional schemes limits the amount of flexibility of the attestation process. Using a fixed timeout based on the assumed network maximum RTT always demands a high number of attestation routine iterations to be computed, and, due to external interference on the communication channel and execution time jitter, still does not guarantee that an honest prover is always able to respond within the time limit. Therefore, we propose a probabilistic attestation approach that allows us to break the single challenge into a series of challenges with shorter timeouts whenever the breakage is expected to be cost beneficial and there is a high probability that at least one challenge in the series will be replied in time.

For instance, suppose that when estimating the network conditions we observe a maximum RTT of x but most RTT samples observed are smaller or equal to $x/2$. In this case, rather than sending a single challenge with a timeout based on x, it is better to send a series of N challenges each of which with a timeout based on $x/2$. We name this procedure an attestation round. Note that a round might finish without the need to send all N challenges. The verifier only sends a new challenge in the series after the current challenge timeout expires with no response. If the prover correctly replies to a challenge in time, then the attestation round is over, and there is no need for the verifier to send the remaining challenges since the prover has already demonstrated its integrity. Likewise, if a prover incorrectly replies to a challenge, before or even after its timeout, the round is also over, and the prover is classified as compromised.

In practice, our approach is similar to performing traditional attestation multiple times in a row using a shorter timeout and stopping either when the prover passes, fails with a wrong response, or when all challenges of the attestation round have been sent. Observe that in this last case we cannot say for sure whether the prover has been compromised or not. There is always a possibility that an honest device will not be able to reply in time to any challenge in a round. Whereas, an adversary trying to fool the attestation process will always reply correctly but after the timeout. Therefore, we can only use this result as an *indication* of compromise. Nonetheless, we can utilize it in conjunction with additional information, such as the outcome of previous rounds, to make a decision.

4.3.1 *Assumptions*

With exception to assuming a maximum network RTT a priori, we share most assumptions made by software-based attestation mechanisms with strict timing conditions: the attestation routine is implemented in a time-optimal way; the verifier cannot be compromised by the adversary and knows the prover's hardware and software configuration; the prover is in communication range of, and can only communicate with, the verifier during attestation; an authentication system is in place, such as radio frequency fingerprint or SRAM PUF; the adversary has full control of the prover's software but cannot modify its hardware.

Ideally, the WAP would perform the role of the verifier. It is in direct communication range with the provers and could easily keep track of the network RTT, it has unlimited energy, and if it gets compromised then essentially the entire network can be considered to be compromised. Furthermore, it could block any outgoing traffic from devices under attestation.

4.3.2 *Preparation Steps*

Before starting an attestation round, the verifier has to (i) collect a sufficient amount of network RTT samples, (ii) find the probability distribution function that best fit these samples, and (iii) use the distribution to calculate the attestation parameters under the desired confidence level. Table 6 summarizes the notation adopted for the remainder of this chapter.

Table 6: Notation summary.

Term	Description
γ	Time of a single iteration of the attestation routine
δ	Maximum RTT allowed
a	Adversary time advantage
c	CPU clock speed
i	Number of iterations
o	Adversary's overhead per iteration of the attestation routine in CPU cycles
N	Maximum number of challenges in a round
$\mathcal{C}$	Number of challenges sent during attestation
$\mathrm{E}[\mathcal{X}]$	Expected value of random variable $\mathcal{X}$
$\mathcal{E}$	Attestation energy consumption
$\mathcal{R}_{\mathcal{T}}(0,\delta)$	RTT observed during attestation
$\mathcal{T}$	Attestation time
$\mathcal{X}_{\mathcal{T}}(a,b)$	Truncated random variable $\mathcal{X}$ within interval $a \leq x \leq b$
E_A	Attestation routine energy consumption
E_P	Prover energy consumption
E_{rx}	Reception energy consumption
E_{tx}	Transmission energy consumption
P_A	Attestation routine power consumption
P_{rx}	Reception power consumption
P_{tx}	Transmission power consumption
T_A	Attestation routine execution time

4.3.3 *RTT Samples Collection*

We gather network RTT samples by making the verifier *ping* the provers sending Internet Control Message Protocol (ICMP) *echo request* and waiting for *echo reply* packets. This can be done in two different ways: either (i) *online*, when we collect new samples every time before performing attestation; or (ii) *offline*, obtaining all samples only once during an initialization phase.

On the one hand, the online collection allows us to estimate the actual network conditions before triggering attestation enabling the use of a different timeout for each round. As a result, if conditions are good, we can decrease the timeout, consequently reducing the number of iterations a prover has to perform. While if conditions are bad, we can increase the timeout, which has the opposite effect but also increases the chances of honest provers replying in time. To achieve this dynamic timeout behaviour we amend the challenge message to include, besides a nonce, the number of iterations to be executed and modify the attestation routine to also use this information as a parameter. In order to find a distribution that best represents the actual network conditions during attestation, we allow the use of a dynamic number of RTT samples. We collect ten samples, find the best fitting distribution, calculate the attestation parameters and repeat this process until the time spent collecting samples is higher than the estimated challenge timeout.

On the other hand, the offline collection benefits from not having to collect samples and compute the attestation parameters every time. In this scenario, a large sample set is preferable as it will be more likely to capture the average channel conditions.

4.3.4 *Network Conditions Estimation*

Having the network RTT samples, the next step in our proposed approach is to find the corresponding best fitting distribution. First we use the Maximum Likelihood Estimation (MLE) method to fit different distributions to the RTT samples. Then, to find the best fitting distribution, we calculate the Sum of Squared Errors (SSE) for each distribution and select the one with the smallest SSE.

Once a distribution is chosen we can use its corresponding Cumulative Distribution Function (CDF) to calculate the probability of successfully sending and receiving the challenge and response messages for any given RTT limit δ. We ACCEPT a response if it is received within δ and REJECT it otherwise. Therefore, we can compute the probabilities of sending two, three, up to N challenges in a row, where at least one of them receives a response within δ according to the equation below:

$$\begin{aligned} P(\textsc{accept}\text{ at least 1 response in } N \text{ attempts} \mid \delta) &= \\ 1 - P(\textsc{reject}\text{ all responses in } N \text{ attempts} \mid \delta) &= \\ 1 - P(\textsc{reject} \mid \delta)^N \end{aligned} \tag{19}$$

4.3.5 *Attestation Parameters*

We can represent the effective number of challenges sent during attestation as a discrete random variable $\mathcal{C} = \{1, 2, 3, ..., N\}$. Both the attestation time and energy consumption depend on the expected value of $\mathcal{C}$, which can be calculated as follows:

$$\mathrm{E}[\mathcal{C}] = \sum_{j=1}^{N} c_j \cdot P(\mathcal{C} = c_j \mid \delta) \tag{20}$$

Where $P(\mathcal{C} = c_j \mid \delta)$ is:

$$P(\mathcal{C} = c_j \mid \delta) = \begin{cases} P(\textsc{reject} \mid \delta)^{c_j - 1} \cdot P(\textsc{accept} \mid \delta) & \text{if } c_j < N \\ P(\textsc{reject} \mid \delta)^{c_j - 1} & \text{if } c_j = N \end{cases} \tag{21}$$

The attestation time depends on $\mathcal{C}$, δ, and on the execution time of the attestation routine T_A, which also depends on δ. We can denote the effective attestation time as a continuous random variable $\mathcal{T}$ defined as:

$$\mathcal{T} = \mathcal{C} \cdot (\mathcal{R}_T(0, \delta) + T_A(\delta)) \tag{22}$$

Where $\mathcal{R}_T(0, \delta)$ denotes a truncated random variable, in the interval $[0, \delta]$, that represents the RTT observed during attestation. We truncate it by δ because once the time limit expires the verifier stops waiting for the response and immediately transmits the next challenge if there is one.

Let us denote the time taken to execute a single iteration by γ, then we can compute T_A as follows:

$$T_A(\delta) = i \cdot \gamma = \frac{(c \cdot \delta)}{o} \cdot \gamma \tag{23}$$

Now, assuming that $\mathcal{C}$ and $\mathcal{R}_T(0,\delta)$ are independent, we can calculate the expected value of $\mathcal{T}$ as:

$$\begin{aligned} \mathrm{E}[\mathcal{T}] &= \mathrm{E}[\mathcal{C} \cdot (\mathcal{R}_T(0,\delta) + T_A)] \\ &= \mathrm{E}[\mathcal{C} \cdot \mathcal{R}_T(0,\delta)] + \mathrm{E}[\mathcal{C} \cdot T_A] \\ &= \mathrm{E}[\mathcal{C}] \cdot \mathrm{E}[\mathcal{R}_T(0,\delta)] + T_A \cdot \mathrm{E}[\mathcal{C}] \end{aligned} \tag{24}$$

Where, given its CDF $F(r)$, its Probability Density Function (PDF) $f(r)$, and the auxiliary function $g(r) = f(r)\ \forall\ 0 \leq r \leq \delta$, the expected value of $\mathcal{R}_T(0,\delta)$ can be computed as:

$$\mathrm{E}[\mathcal{R}_T(0,\delta)] = \frac{\int_0^\delta r \cdot g(r)\,\mathrm{d}r}{F(\delta) - F(0)} \tag{25}$$

The prover's energy consumption E_P for a single challenge can be calculated as:

$$E_P = E_{rx} + E_A + E_{tx} \tag{26}$$

Where the challenge reception energy consumption E_{rx} can be computed as:

$$E_{rx} = \frac{\text{Packet size}}{\text{bit rate}} \cdot P_{rx} \tag{27}$$

And, similarly, the response transmission energy consumption E_{tx}:

$$E_{tx} = \frac{\text{Packet size}}{\text{bit rate}} \cdot P_{tx} \tag{28}$$

While the attestation routine energy consumption E_A follows:

$$E_A = T_A \cdot P_A \tag{29}$$

With these equations we can define the prover's effective energy consumption as a discrete random variable $\mathcal{E}$:

$$\mathcal{E} = \mathcal{C} \cdot E_P \tag{30}$$

And its expected value:

$$\mathrm{E}[\mathcal{E}] = \sum_{j=1}^{N} c_j \cdot E_P \cdot P(\mathcal{C} = c_j \mid \delta) \tag{31}$$

4.3.6 *Attestation Round*

Once the attestation parameters are defined, a verifier can start an attestation round that may consist of a series of challenges which respect the following specification:

1. All challenges in a round have the same timeout, and thus, require the same number of iterations to be computed by the prover.
2. Each challenge in the series has a different nonce from the previous ones. Otherwise, an adversary would receive a challenge, solve it expiring its timeout, but then receive it again having the correct answer already computed and would be able to respond in time.
3. The verifier sends a new challenge of the series immediately after the timeout of the current challenge expires without the reception of a response.
4. Whenever a challenge is correctly responded within its timeout, the round is concluded, and the prover is considered to be honest.

5. Whenever a challenge is incorrectly responded, within or after its timeout, the round is concluded, and the prover is considered to be compromised.

4.3.7 *Security*

To analyse the security of the proposed scheme we examine how it relates to existing attacks on attestation previously described in the literature (covered in Chapter 2 Section 2.1.2).

By using a cryptographically secure pseudo-random number generator to produce a different nonce for each challenge both the *precomputation* and *replay* attacks are no longer feasible.

The *collusion*, *impersonation*, and *proxy* attacks are not valid under the adopted threat model. As with previous schemes, we assume the prover can only communicate with the verifier during attestation, which prevents both the *collusion* and *proxy* attacks, and that some authentication mechanism is in place, which prevents the *impersonation* attack. Nevertheless, our scheme reduces the success chances of these attacks. Since we use shorter timeouts it becomes harder for nodes to exchange or relay messages and still reply to the verifier in time. Furthermore, the use of shorter timeouts reduces execution time jitter, thus also preventing the *high execution time variance* attack.

The *forgery*, *memory copy*, *data substitution*, *compression*, and *return-oriented programming* attacks aim to overcome the attestation routine itself. While these attacks are implementation specific, our scheme can be used with different implementations. We only require the routine to also take the number of iterations as a parameter, which has no security implications. In traditional approaches, since the timeout is fixed, the adversary always knows the number of iterations, while in our scheme the adversary only gets to know this information when it receives the challenge. From the attestation routine perspective, the fact that we may execute the routine multiple times in a row cannot be exploited since each run is independent of the previous. There is however, a new attack that an adversary could try to perform against our scheme, which we describe below:

Delay attack. Knowing that we estimate the network conditions with RTT samples, an adversary may deliberately delay all messages handled by compromised nodes. In doing so, the adversary can influence our estimations and increase the calculated challenges timeout in an attempt to gain more time to perform extra computations.

Note, however, that whenever we increase a challenge timeout we also increase the number of iterations of the attestation routine to compensate for any time advantage the adversary may have, as shown in Equation 18 and Figure 15. Therefore, the *delay* attack would be ineffective in overcoming attestation. In the worse case scenario, it would force the use of a large timeout even though the network conditions are good, similar to what happens with traditional attestation approaches. Nevertheless, a possible countermeasure to this attack is to collect RTT samples from multiple provers and apply anomaly detection techniques to exclude the malicious data introduced by the attacker.

Ultimately, our approach does not make attestation vulnerable to any existing or foreseeable new attacks. In reality, the use of shorter timeouts hinders the execution of *collusion*, *impersonation*, *proxy*, and *high execution time variance* attacks.

4.4 ANALYTICAL EVALUATION

To compare results, we analyse the performance of our proposal using the RTT samples from the same channel we used to discuss traditional approaches in Section 4.2 — P2P PIN ch. 6 in Figure 14 (b).

As seen in Section 4.3.4, once we have the network RTT samples, we need to find the probability distribution that best fits our data. Figure 16 shows the PDF of the top five fitting distributions over the RTT samples. In this example the Exponential Power Distribution (labeled exponpow) is our choice, as it is the one with the smallest SSE.

Having selected a distribution, we can then compute the probability of sending N challenges and receiving at least one response within a given timeout. Figure 17 shows the results of Equation 19 for $N = \{1, ..., 10\}$. It is possible to notice that the higher the number

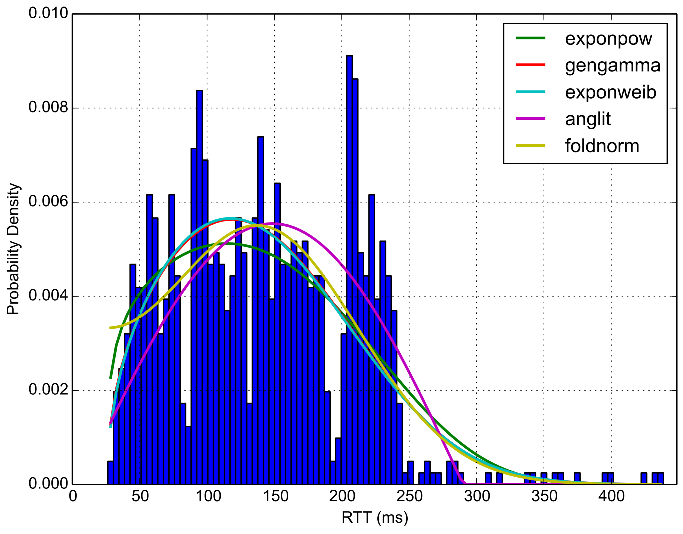


Figure 16: Best fitting distributions for the RTT samples.

of challenges allowed, the faster the probability of success grows. Which is intuitive, the more messages we send, the higher is the chance that at least one of them will be replied in time.

4.4.1 *Time and Energy*

Figure 18 shows the maximum and expected attestation time when determining δ under different confidence levels. It is possible to see that the higher the confidence level, the higher is $\mathcal{T}_{max}$. The reason is that $\mathcal{T}$ increases as δ increases, and higher confidence levels require higher δ. On the other hand, as N increases, $\mathrm{E}[\mathcal{T}]$ decreases until it reaches a minimum and then starts to grow. The explanation is that as N increases δ decreases, so it is possible for the prover to reply earlier. However, when δ gets too small, it is more likely that the prover will not be able to respond in time. Thus, more challenges are required and the total expected time starts to increase.

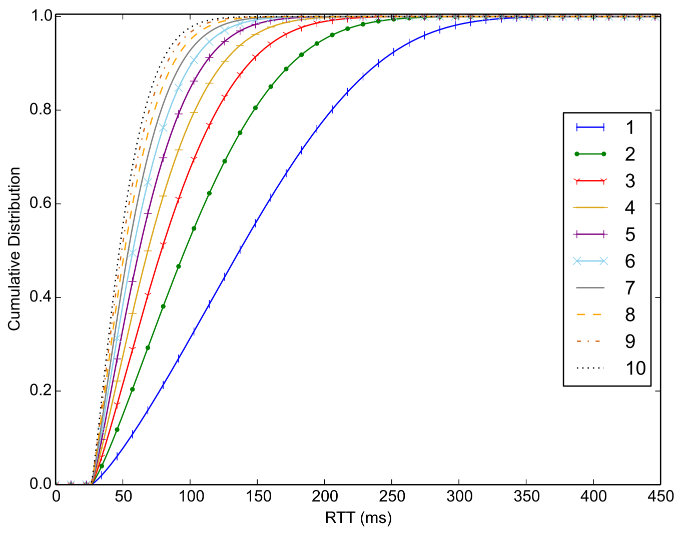


Figure 17: Probabilities of sending from 1 to 10 challenges in a row where at least one is replied within δ.

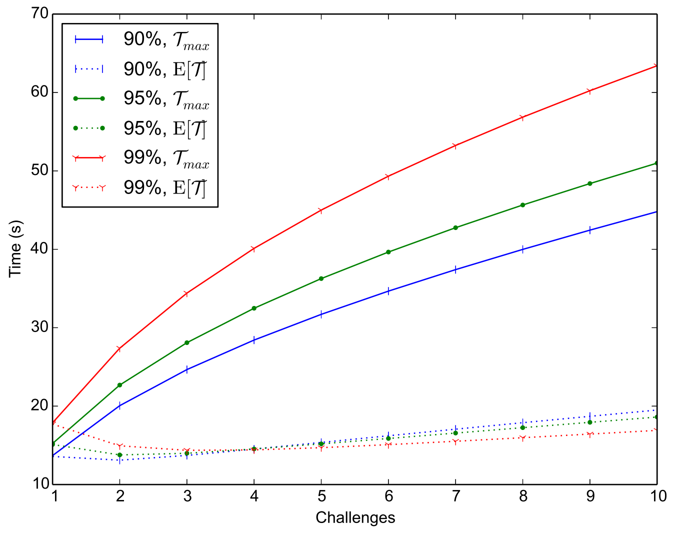


Figure 18: Maximum and expected attestation time under different confidence levels.

Table 7: Edison power consumption (mW).

	Power Management	
	On	Off
P_{Tx}	535	716
P_{Rx}	604	649
P_A	714	839

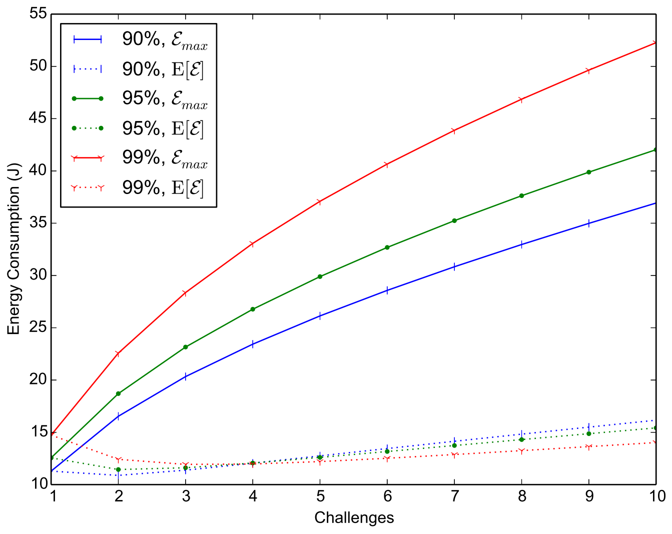


Figure 19: Prover's maximum and expected energy consumption (power management off).

To calculate the energy consumption, we have measured the power consumption of the Intel Edison and observed the values shown in Table 7. In our implementation, the packet size, 90 bytes, is the same for both challenge and response. It includes the 802.11 data frame (38 bytes), the IP header (20 bytes), ICMP header (8 bytes), and payload (24 bytes). The channel bit rate is 54 Mbit/s. Figure 19 shows the prover's maximum and expected energy consumption with its Wi-Fi driver power management turned off. The same analysis we did for the attestation time is valid here. The higher the confidence level, the higher is $\mathcal{E}_{max}$, and as N increases E[$\mathcal{E}$] decreases until it reaches a minimum and then starts to grow.

4.4.2 *Summary*

Since both $\mathrm{E}[\mathcal{T}]$ and $\mathrm{E}[\mathcal{E}]$ reach minimum values, we can set N to optimize either the attestation time or energy consumption. Furthermore, to find their minimums we do not need to compute for arbitrary values of N. Since both $\mathrm{E}[\mathcal{T}]$ and $\mathrm{E}[\mathcal{E}]$ continually decrease as N increases until they reach a minimum, we can stop the computation whenever we find the turning point.

Table 8 summarizes how $E[C]$, i, and T_A change under the same N by the use of different confidence levels to estimate δ. We can also use the values from this table to compare our performance against the one we analysed in Section 4.2. While traditional attestation schemes are limited to the maximum RTT estimated a priori our approach enables a wide range of options for defining the number of iterations and timeout. We demonstrated that for these network conditions traditional approaches would require 211463783 iterations to be performed by the attestation routine, which would take approximately 24.6 seconds to execute plus the RTT to send the challenge and receive the response. In our best scenario, we could choose $N = 10$, with 90% confidence level, and have our first challenge in the series be responded within the timeout. This would take only 4.4 seconds plus the messages RTT. Even if we limit ourselves to $N = 1$, instead of using the worst RTT measured, we can with 99% confidence level reduce the attestation routine execution time to 17.6 seconds.

4.5 EXPERIMENTAL EVALUATION

To experiment with real devices, we have implemented an honest and a malicious version of the attestation routine, as described in Section 4.1, for the Intel Edison platform. Note that the malicious version simulates an attacker trying to bypass attestation without being detected. At the current stage, our implementation for the Intel Edison is still a prototype. For instance, although the Edison is a multi-core platform, currently our implementation

Table 8: Parameters according to confidence level.

N	%	$E[\mathcal{C}]$	δ(ms)	i	T_A(s)
	90	1.000	239.939	115660904	13.455
1	95	1.000	266.945	128679161	14.970
	99	1.000	313.216	150983554	17.564
	90	1.316	175.723	84706215	9.854
2	95	1.224	198.747	95804546	11.145
	99	1.100	239.939	115661161	13.455
	90	1.680	144.025	69426404	8.077
3	95	1.504	164.056	79082148	9.200
	99	1.262	200.983	96882741	11.271
	90	2.056	124.506	60017477	6.982
4	95	1.802	142.263	68577185	7.978
	99	1.448	175.724	84706473	9.854
	90	2.439	111.091	53550469	6.230
5	95	2.108	127.071	61253624	7.126
	99	1.645	157.687	76011791	8.843
	90	2.824	101.226	48795302	5.677
6	95	2.417	115.778	55810064	6.493
	99	1.848	144.025	69426501	8.077
	90	3.211	93.628	45132632	5.250
7	95	2.729	107.007	51581855	6.001
	99	2.054	133.255	64234759	7.473
	90	3.598	87.574	42214613	4.911
8	95	3.041	99.973	48191165	5.606
	99	2.262	124.507	60017824	6.982
	90	3.987	82.624	39828466	4.633
9	95	3.355	94.187	45402390	5.282
	99	2.472	117.240	56514707	6.575
	90	4.376	78.493	37836967	4.402
10	95	3.670	89.338	43064981	5.010
	99	2.683	111.091	53550622	6.230

executes on a single core. This is by no means a limitation imposed by our scheme. In fact, our proposed stochastic approach can be applied to other platforms and to different attestation routines, as long as the latter can take the number of iterations as a parameter (only necessary for the online configurations). Nevertheless, our proof of concept is enough to show the feasibility of our approach.

4.5.1 *Settings*

We designed our stochastic attestation to work either with online or offline RTT samples collection, as described in Section 4.3.3. To provide a fair comparison with the current state of the art solution, we have also implemented an online and offline version of the traditional single challenge-response protocol, even though works in the literature only take the offline approach. Thus, we test four different configurations: our proposed *stochastic online* and *stochastic offline* solutions, the current state of the art solution ([44, 37, 122, 38]) *maximum offline*, and, for completeness, an adaptation of it which we name *maximum online*.

We performed our experiments in three different locations using real-world Wi-Fi networks: (i) the already mentioned Imperial College Central Library; (ii) a residential building where different apartments share a communal Wi-Fi; and (iii) a coworking office space. Once again, we emphasize we had no control over the ongoing network traffic in any of these locations.

We used Wireshark to capture the packets exchanged between the prover and the verifier, which allowed us to picture missing packets and responses out of bound. In addition, by putting the wireless card in monitor mode, we were able to observe all wireless traffic during attestation rounds, which allowed us to tell how noisy each location was during our tests. By checking MAC addresses, we have also counted the total number of distinct WAP and Ad-hoc connections in range and unique devices observed during our experiments — note that these numbers were not constant. Table 9 summarizes our findings. It is interesting to see that even though the library was by far the noisiest environment, it was

Table 9: Observed wireless conditions.

	Building	Library	Office
Average bytes/s	27.7 K	2.67 M	23 K
Connections in range	26	213	118
Min. RTT (ms)	2.33	2.83	2.69
Max. RTT (ms)	985.76	695.13	389.56
Unique devices	625	4203	2111

the building location which presented the highest observed RTT. We also noticed that the Edison devices were getting disconnected from the library WAP every time the attestation routine took more than 34 seconds to execute. Whenever this happened, the device would need to first reconnect to the WAP and only then send the attestation response, thus elapsing the timeout. Therefore, we had to limit the maximum RTT allowed to 600 ms to bound the attestation routine execution time and consequently avoid this from happening. The reason for the disconnections was most likely a configuration in the library WAP as the same did not happen in the building location. Effectively, this is another limitation of using the maximum RTT to define the attestation timeout given that the stochastic approach is much less likely to use long timeouts.

In each location, we first evaluated the *online* configurations and then used the RTT samples they collected to evaluate the *offline* configurations. To find the best fitting distribution for the samples collected we use the Scipy library, which has more than eighty distributions. To reduce the computation time, we experimented and found a subset of distributions which performed better than others[2]. We then calculate the attestation parameters with 99% confidence level and select the maximum number of challenges in a round (N) with the smallest expected energy consumption ($\mathrm{E}[\mathcal{E}]$). On average, the *stochastic online* configuration collected 16.5 RTT samples per attestation round, which took around 6.4 seconds and spent roughly more 2 seconds to find the best fitting distribution and compute the attestation parameters. Whereas, the *maximum online* configuration did not

2 Those were: alpha, foldcauchy, halfcauchy, powerlognorm, fatiguelife, lomax, johnsonsu, cauchy, and gilbrat.

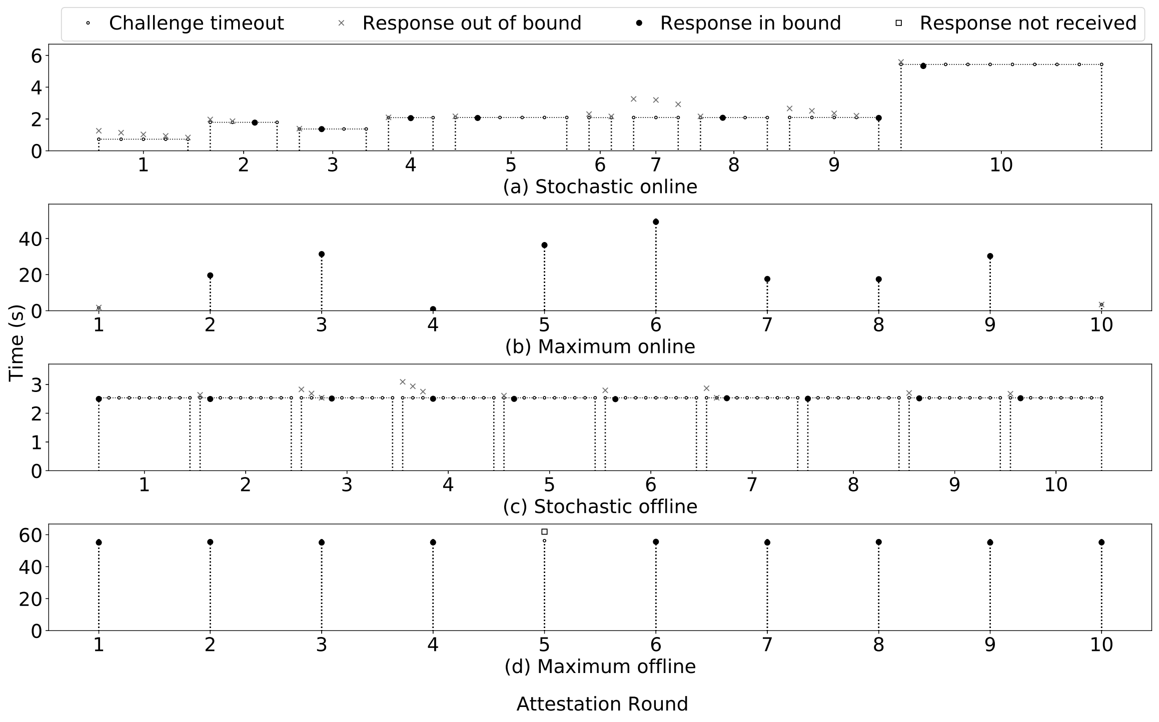


Figure 20: Interaction pattern of the different approaches.

need to fit a distribution, but rather simply use the maximum RTT observed. Thus, we used a fixed RTT sample of size ten, which took on average 3.6 seconds to collect.

For all locations we attested both the honest and malicious devices fifty times each. To better illustrate how each configuration behaves, Figure 20 shows the first ten attestation rounds performed in the building location. It is possible to see how the maximum number of challenges and their timeout change from round to round in *online* configurations, which is a reflection of the network conditions changing over time. Whereas the *offline* configurations use the same parameters for all rounds. We explicitly show all the challenges that could be sent during an attestation round by plotting their timeout. This enables us to visualize how many other chances the prover still had when it passed attestation.

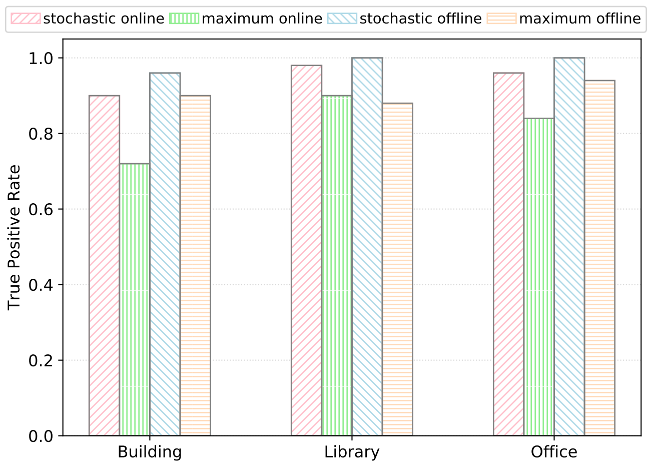


Figure 21: Attestation TPR achieved under different environments.

4.5.2 *True Positives / False Positives Results*

Figure 21 shows the percentage of times the honest device was able to pass attestation, in other words, the True Positive Rate (TPR), achieved by each configuration in all locations. As can be seen, the *stochastic offline* configuration achieved the best results in every location with an average of 99%, followed by the *stochastic online* (95%), *maximum offline* (91%), and lastly *maximum online* (82%). All configurations presented a 0% False Positive Rate (FPR), meaning that not even once the malicious prover was able to pass attestation. This is expected, since we know the overhead of the adversary attack and set the number of iterations accordingly (Equation 18), such that a malicious prover can only respond after the timeout expires. The only way a false positive could happen in this scenario is if we had a sufficiently high execution time jitter allowing the attacker to reply within time occasionally. This did not occur in our experiments.

4.5.3 *Time and Energy Consumption*

Figure 22 displays the attestation time measured by the verifier and energy consumption of the honest prover for all configurations and locations. Bear in mind that it does not take into account the time and energy spent collecting RTT samples, which is a disadvantage of *online* configurations. It depicts the minimum, maximum, average (where the error bar shows the standard deviation), and total (sum of all fifty attestation rounds) values observed during our experiments. Note how small is the standard deviation of the *maximum offline* configuration in comparison to the others. That is because it is the only configuration that does not make use of either multiple challenges or different timeouts per attestation round. Thus, the only things changing per round are the attestation routine execution time jitter and the actual challenge-response RTT. It is also possible to see that the *stochastic offline* configuration outperformed the others (except for the *stochastic online* in the library) presenting the best overall results with an aggregate (all three locations) time of 845.55 s and energy consumption of 648.45 J, followed by the *stochastic online* (1025.08 s and 815.98 J), *maximum online* (1616.54 s and 1350.37 J), and *maximum offline* (5548.35 s and 4646.87 J).

Figure 23 shows the attestation time and energy consumption of the malicious prover, again not considering the time and energy consumed collecting RTT samples. In this case, the *stochastic offline* configuration also has a tiny standard deviation. That is because the malicious prover always fails all challenges in a round and all rounds use the same parameters. Note that we are assuming an adversary that tries to bypass attestation by sending a correct answer. If the compromised device were to send a wrong answer, then it would be detected and the attestation rounds would end sooner. This time the *maximum online* configuration outperformed the others in every location presenting the best results (1963.65 s and 1618.64 J), while the *stochastic offline* is the second best (2975.28 s and 2239.85 J), *stochastic online* third (3524.89 s and 2747.35 J), and *maximum offline* fourth (5637.77 s and 4646.87 J).

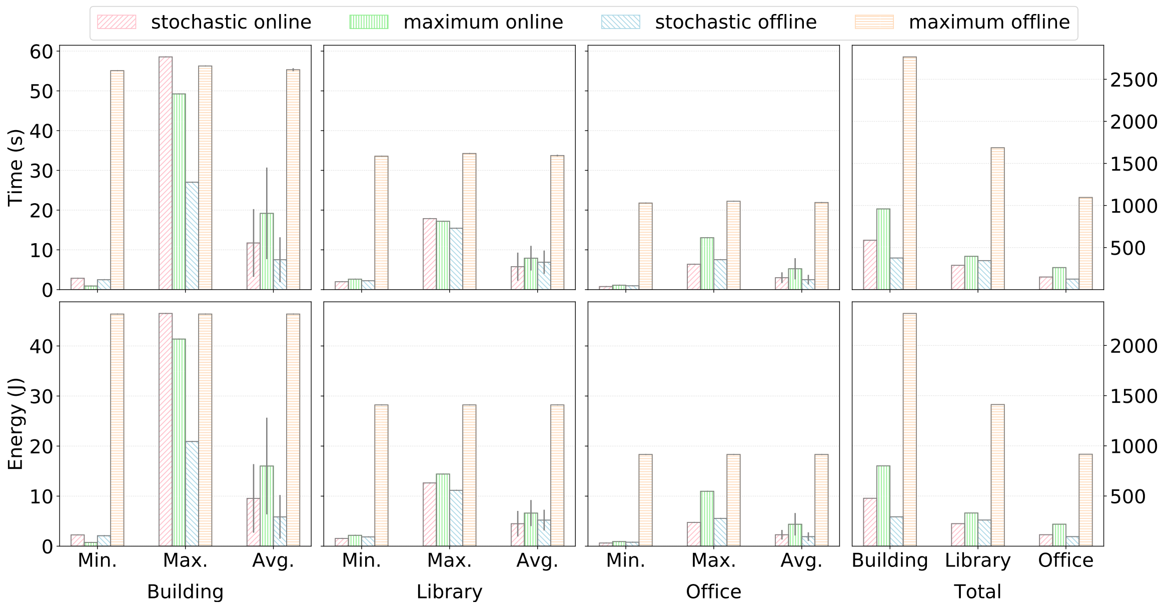


Figure 22: Attestation time and energy consumption (honest prover).

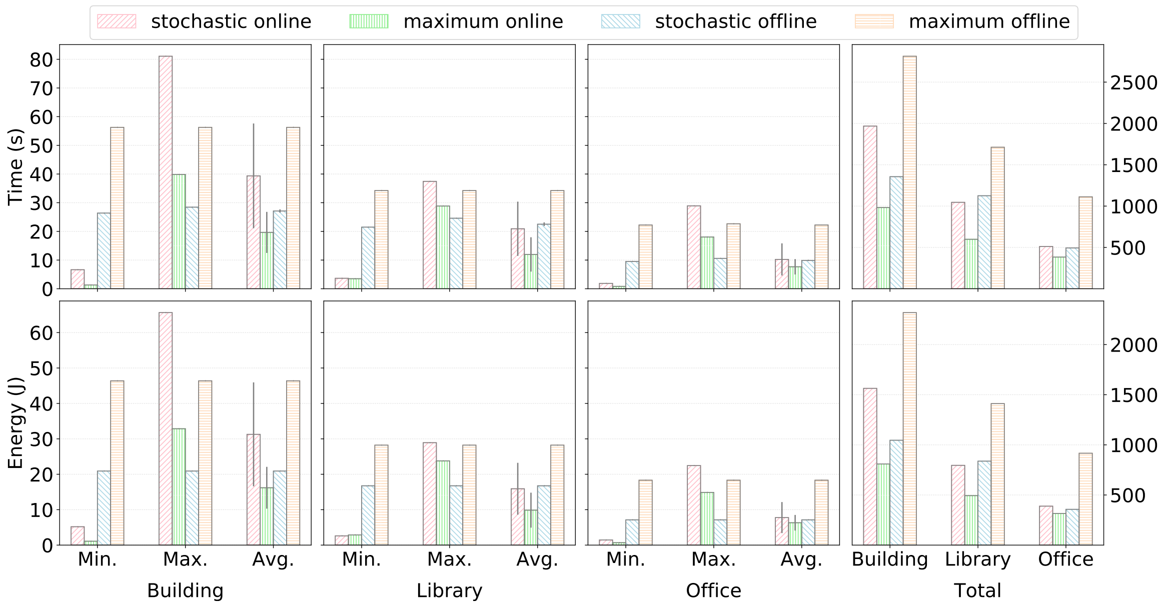


Figure 23: Attestation time and energy consumption (malicious prover).

4.5.4 *Summary*

It is worth mentioning that we expect the majority of devices in a network not to be compromised, and if that is not the case, then the time and energy spent attesting malicious devices is certainly not a primary concern. Therefore, reducing the attestation time and energy for honest devices is much more valuable than doing the same for malicious ones.

In summary, the *stochastic offline* configuration presented the best overall results. In comparison to the current state of the art solution (*maximum offline* configuration), it presented an 8% TPR increase in detecting honest provers and reduced the attestation time and energy consumption around seven times for honest devices and two times for malicious ones. The considerably lower TPR achieved by the *maximum online* configuration is reason enough to disregard it as a good option. Finally, the *stochastic online* configuration is a valid second choice, but just not as good, especially when we consider the extra time and energy required to collect RTT samples before each attestation round.

4.6 RELATED WORK

To the best of our knowledge He et al. [123] and Yang et al. [19] are the only other software-based attestation works that target the network RTT, with the later being an extension done by the same group of authors. Their approach differs from ours as they propose a multi-hop attestation scheme, while our scheme performs only single-hop attestation. To attest a remote device, the verifier sends a challenge that is relayed across the network until it reaches the prover. The prover's response has to go back through the same path as the challenge. Each relay node records the time when they receive the challenge and response and report this information to the verifier. The verifier then can estimate the average single-hop RTT and detects compromised nodes by using a Bayesian classifier. We identify three drawbacks of this scheme. First, since both the challenge and response have to go through the same path the approach does not tolerate neither node failures or node mobility. Second, to prevent a compromised relay node from modifying the reports of other nodes, they assume the use of cryptographic keys to protect transmitted messages. However, software-based attestation mechanisms are explicitly designed for devices that do not possess any hardware capable of protecting such keys. Third, they assume that all hops in the network have a similar RTT. This is not realistic since each hop may suffer from different sources of interference. Furthermore, the experiments we realized in this chapter have shown that even a single-hop may drastically change its conditions over time.

4.7 CONCLUSIONS

In this chapter, we present a novel way of performing attestation which is a step further in the development of practical software-based mechanisms. Instead of using the maximum known network RTT to send a single large attestation challenge, which demands a high number of computations to be performed, we use a series of short challenges that require fewer computations and can be finalized whenever a correct response is given in time. As can be seen from our experimental results, when compared to the current state of the art solution (*maximum offline* configuration) our proposal (*stochastic offline* configuration) considerably reduces the overall attestation time and energy consumption (around seven times for honest devices and two times for malicious ones) while improving the detection rate of honest devices (8% higher TPR) without compromising its security (0% FPR). Nevertheless, all previous proposals can be adapted to follow this new interaction pattern and benefit from its advantages.

5 MODELLING THE NETWORK HEALTH

In the previous chapters, we looked into how security techniques such as attestation and measurements inspection can identify compromised nodes in a network. However, just knowing which devices have been compromised is not enough for us to understand how the network as a whole is affected. The health of a network is not a binary property indicating whether it has been compromised or not, but rather an indicator of how well it can operate in its current state and fulfil its functions. Moreover, one can view the impact of an attack from different perspectives, for instance, the effects it has on the confidentiality, integrity, or availability of the data and services provided by the network. In this chapter, we focus on the latter.

In particular, we view availability as a connectivity problem. Once we detect a node has been compromised, we know we can no longer trust any data it reports. However, we also have to consider that at any point a compromised device can stop to relay data from other nodes that need it to reach the sink or vice-versa. For instance, a malicious node may not forward commands from the base station to trigger a node's actuator when an event is detected. Besides, it makes perfect sense for a node injecting measurements not to relay data from honest sensors so that a measurements inspection mechanism has fewer chances of detecting the attack. Therefore, the network has to keep operating without relying on compromised nodes, as if they were disconnected from it.

To understand the effects this has on the network we propose a mathematical model to represent the health of WSNs. Our model combines the knowledge regarding compromised nodes with additional information that quantifies the importance of each node. More specifically, we define the significance of a node according to the tasks it performs and the role it plays on the network connectivity. While the value of each task is intrinsically dependent on the network application, the importance of a node for the network connectivity is generalizable and depends mainly on its topological position. Intuitively, a node that forwards traffic of a few nodes in the network should be less important than a node that forwards traffic from one dense network segment to another.

In network theory, several centrality measures have been proposed to identify the most important nodes in a network according to its topology, e.g., degree centrality, closeness centrality, betweenness centrality, among others [25]. Nevertheless, no ultimate centrality measure suits all applications, and it is usually the case that the optimal measure for a given scenario is not the best for a different one. For instance, betweenness centrality only considers communication along shortest paths. Conversely, current flow betweenness centrality assumes information flows like an electrical current across the entire network. However, WSN traffic usually originates or terminates at the sink node and does not necessarily go through the shortest paths or spread across all network nodes. Therefore, we propose a new metric, named *current-flow sink betweenness*, and evaluate which metric is more suitable to recognize the impact each node has on the connectivity of WSNs. As of today, no other work has explored this issue in depth.

The rest of this chapter is organized as follows. In Section 5.1 we introduce preliminary concepts and formalise our network health model. Then, in Section 5.2 we describe common centrality measures as well as the proposed current-flow sink betweenness measure. Afterwards, in Section 5.3, we conduct a comprehensive performance analysis among all of the described measures to evaluate their adaptability and suitability in the context of WSNs and instantiate our proposed health model using the best-suited metric. In Section 5.4 we discuss related work. Finally, in Section 5.5 we present our conclusions and future work directions.

5.1 NETWORK HEALTH

Our concept of network health refers to the ability of a network to properly perform its functions. A network operates at maximum health when all of its nodes are working correctly. As nodes fail or are even attacked, the health of the network degrades until it becomes fully dysfunctional. Moreover, distinct nodes might have a different impact on the network operation. We here propose a model that captures this notion and is capable of expressing the health of WSNs into a single value.

5.1.1 *Preliminary Concepts and Notation*

Let $W = (V, E)$ be a WSN modelled as an undirected graph where V represents the set of sensors and E corresponds to the set of connectivity edges among nodes. An edge $(v_i, v_j) \in E$ expresses the fact that node v_i is within communication range of node v_j. This relationship is symmetric, meaning edges (v_i, v_j) and (v_j, v_i) are identical. The *order* of the graph is $|V|$, the number of nodes in the network. Whereas the *size* of the graph is $|E|$, the number of edges.

Within a WSN, a path $p = (V', E') \subseteq W$ is a non-empty subgraph of W where $V' = \{v_0, v_1, \ldots, v_k\}$ is a sequence and for each vertex pair $v_i, v_{i+1} \in V'$, $i \in [0, k-1]$, there exists an edge $(v_i, v_{i+1}) \in E' \subseteq E$ that links them.

Note that to represent the network as a graph, we first need to know its topology. We assume it is either known or can be retrieved post-deployment by using a topology or location discovery algorithm [124, 125, 126, 127, 128].

Finally, we assume a sink-to-sensors/sensors-to-sink communication model. We denote the sink node, also known as base station, by $\mathcal{S}$ and the remaining nodes in the network as V^*, such that $V^* = V - \{\mathcal{S}\}$. Furthermore, to simplify our analysis, we can reduce the case with multiple base stations to a scenario with a single one. For that, we introduce an

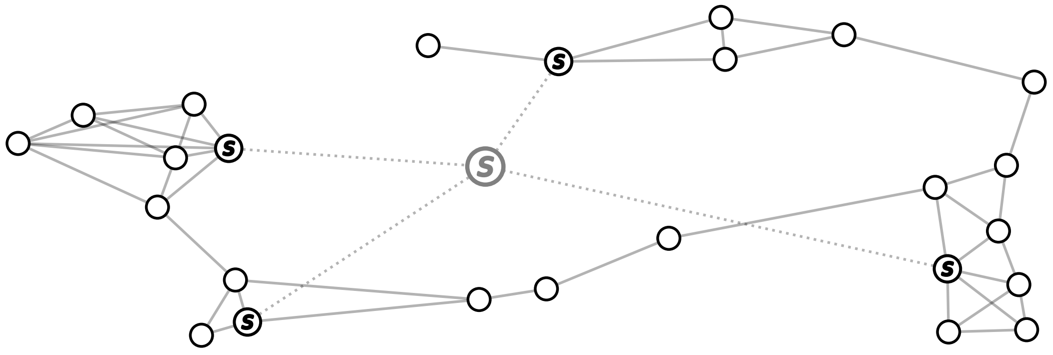


Figure 24: Adding an artificial node to deal with multiple base stations.

artificial node to our graph representation of the network and link it to each of the original sink nodes as shown in Figure 24.

5.1.2 *Model*

We specify the *health* of a WSN as a combination of two main aspects: (i) the importance of each node to the network operation, and (ii) whether or not the node in question is contributing to the network operation. In particular, a node contributes to the network operation if and only if it is *functional* and there is a *safe path* connecting it to the sink. However, a node that is either not functional or that is disconnected from the sink causes damage to the network proportionally to how important such node is.

In this context, we formally define the health of a network $W = (V, E)$ as:

$$\mathcal{H}(W) = 1 - \frac{\sum\limits_{v_i \in V^*} I(v_i) \cdot \Big(1 - \big(F(v_i) \cdot S(v_i)\big)\Big)}{\sum\limits_{v_i \in V^*} I(v_i)}$$

where $I(v_i)$ quantifies the importance of node v_i, $F(v_i)$ is a boolean function indicating if node v_i is functional, and $S(v_i)$ is a boolean function indicating if there is a *safe path* connecting v_i to the sink.

Every node v_i is considered to be functional ($F(v_i) = 1$) until either it fails (e.g., runs out of energy) or is classified as compromised by the security mechanism in place. In such cases, $F(v_i)$ will return 0.

We define a *safe path* as a path where all nodes involved are functional. If there is such a path connecting node v_i to the sink then $S(v_i)$ outputs 1, otherwise 0. It is worth mentioning that computing all paths when looking for the existence of a safe path is too expensive and not a viable solution. Instead, we take an alternative approach. First, we remove all compromised nodes, with their corresponding edges, from our graph representation. Then we compute the connected components in the resulting graph. If there is only one such component, then we know there is a safe path. Otherwise, we check if v_i is in the sink component. If it is not, then there is no safe path connecting the two.

Although it is typically assumed that the sink node has an unlimited amount of energy and cannot be compromised, for completeness, it is possible to extend our health definition to consider such scenario:

$$\mathcal{H}'(W) = F(\mathcal{S}) \cdot \mathcal{H}(W)$$

Note that our health definition always returns a value in the interval $[0, 1]$, where 0 denotes a completely dysfunctional network and 1 denotes the network is fully functional. This means that the fraction being subtracted in the equation is effectively the damage sustained by the network.

We quantify the importance of each node based on the tasks it executes as well as its effect on the network connectivity. While different nodes might execute distinct tasks, such as monitoring dissimilar physical phenomena, working as an actuator, among others, the value of each task is application specific. For simplicity, we assume that all nodes in the network perform the same tasks or tasks of equivalent values. In this case, a node's importance is defined solely by how it may affect the network at a connectivity level, which is defined primarily based on its topological position. To this end, we can use centrality measures to identify which nodes are more important. For the remainder of this chapter we investigate which metric is more suitable to our requirements.

5.2 CENTRALITY MEASURES

Several centrality measures have been proposed in the literature, and while they all try to identify the most important nodes in a network, each metric has a different concept of what makes a node important. In this section, we give an overview of the most traditional metrics as well as some of the more recently proposed ones, and also introduce a new measure called *current-flow sink betweenness*. While we describe the main intuition behind each measure and how to compute them, we refer the reader to their original work for further details.

5.2.1 *Degree Centrality*

Degree centrality measures the number of direct connections a node has with other nodes in the network [129]. Formally:

$$C_D(v_i) = \sum_{v_i \neq v_j} e(v_i, v_j)$$

where $e(v_i, v_j)$ is 1 if there is an edge connecting v_i to v_j and 0 otherwise.

5.2.2 *Closeness Centrality*

Closeness centrality quantifies how close, on average, a node is to all other nodes in the network [129]. It is defined as:

$$C_C(v_i) = \frac{1}{\sum\limits_{v_i \neq v_j} d(v_i, v_j)}$$

where $d(v_i, v_j)$ is the distance — number of edges in a shortest path — between v_i and v_j. Note that if there is no path between two nodes, the distance is infinite by convention. Thus, this metric is not suitable in such scenarios.

5.2.3 *Harmonic Centrality*

With the same goal of closeness centrality, but still applicable to disconnected graphs, harmonic centrality is defined as the sum of the inverted distances rather than the inverted sum of distances [130]. Formally:

$$C_H(v_i) = \sum_{v_i \neq v_j} \frac{1}{d(v_i, v_j)}$$

where $1/\infty$ (when there is no path from v_i to v_j) is 0.

5.2.4 *Betweenness Centrality*

Betweenness centrality measures the number of times a node appears on the shortest paths between all pairs of nodes in the network [129]. It is formally defined as:

$$C_B(v_i) = \sum_{v_i \neq v_j \neq v_k} \frac{\sigma(v_j, v_k | v_i)}{\sigma(v_j, v_k)}$$

where $\sigma(v_j, v_k)$ is the total number of shortest paths from v_j to v_k and $\sigma(v_j, v_k|v_i)$ is the number of those paths that pass through v_i.

5.2.5 *Current-flow Betweenness Centrality*

Whereas betweenness centrality assumes information flows over shortest paths, current-flow betweenness centrality adopts a model where information spreads as an electrical current [131]. Formally:

$$C_{CB}(v_i) = \sum_{v_i \neq v_j \neq v_k} \tau(v_j, v_k | v_i)$$

where $\tau(v_j, v_k|v_i)$ is the amount of current that flows through v_i when a unit of current is injected at vertex v_j and extracted at vertex v_k. This metric is also known as random-walk betweenness centrality.

5.2.6 *Current-flow Closeness Centrality*

While closeness centrality measures distance, current-flow closeness centrality quantifies the absolute potential differences among the nodes in the network [132]. It is defined as:

$$C_{CC}(v_i) = \frac{1}{\sum\limits_{v_i \neq v_j} \hat{p}(v_i, v_j | v_i, v_j)}$$

where $\hat{p}(v_i, v_j|v_i, v_j)$ quantifies the effective resistance between v_i and v_j when a unit of current is injected at vertex v_i and extracted at vertex v_j. This metric is equivalent to information centrality [133].

5.2.7 *Eigenvector Centrality*

Whilst degree centrality simply counts the number of connections of a node, eigenvector centrality assigns proportional scores to a node based on the scores of its neighbours [134]. Formally:

$$C_E(v_i) = \frac{1}{\lambda} \sum_{v_i \neq v_j} e(v_i, v_j) C_E(v_j)$$

where λ is a constant. Given the adjacency matrix A of the network, such that $a_{i,j} = e(v_i, v_j)$, and defining a vector $x = (C_E(v_1), C_E(v_2), \ldots)$ one can rewrite the previous equation as:

$$Ax = \lambda x$$

which allows us to observe that x is the eigenvector of A with eigenvalue λ.

5.2.8 *Katz Centrality*

Katz centrality also defines the score of a node based on its neighbours and can be viewed as a generalization of the eigenvector centrality [135]. It is defined as:

$$C_K(v_i) = \alpha \sum_{v_i \neq v_j} e(v_i, v_j) C_E(v_j) + \beta$$

where α is an attenuation factor penalizing the contribution of distant nodes while β can give extra weight to immediate neighbors.

5.2.9 *Sink Betweenness Centrality*

Sink betweenness is an adaptation of betweenness centrality explicitly designed for WSNs [136]. Since the communication in such networks is usually between sensor nodes and the sink, instead of considering all pair of nodes, this metric only considers the shortest paths between each node and the sink. It is formally defined as:

$$C_{SB}(v_i) = \sum_{v_i \neq v_j \neq \mathcal{S}} \frac{\sigma(v_j, \mathcal{S} | v_i)}{\sigma(v_j, \mathcal{S})}$$

where $\mathcal{S}$ is the sink node.

5.2.10 *Current-flow Sink Betweenness Centrality*

In the same way sink betweenness adapts betweenness centrality, we here propose to adapt the current-flow betweenness centrality to only take into account the paths terminating at the sink. Formally:

$$C_{CSB}(v_i) = \sum_{v_i \neq v_j \neq \mathcal{S}} \tau(v_j, \mathcal{S} | v_i)$$

5.2.11 *Discussion*

Figure 25 illustrates how each of the metrics presented in this section views the importance of the nodes in a network. As can be seen, different metrics have distinct concepts of what makes a node important and can, therefore, lead to diverging results, which means that no metric suits all cases and the optimal measure in a given scenario is usually not the best for a different one.

One limitation of centrality measures is that they do not necessarily specify the *relative importance* between nodes. While nodes are ranked from most to least important, a node with a score of 9 may not be three times more important than a node with a score of 3. Furthermore, the characteristics used to identify the most important nodes do not always generalize to the remaining nodes in the network. In such cases, aside from the most important nodes, the rankings might be meaningless [137].

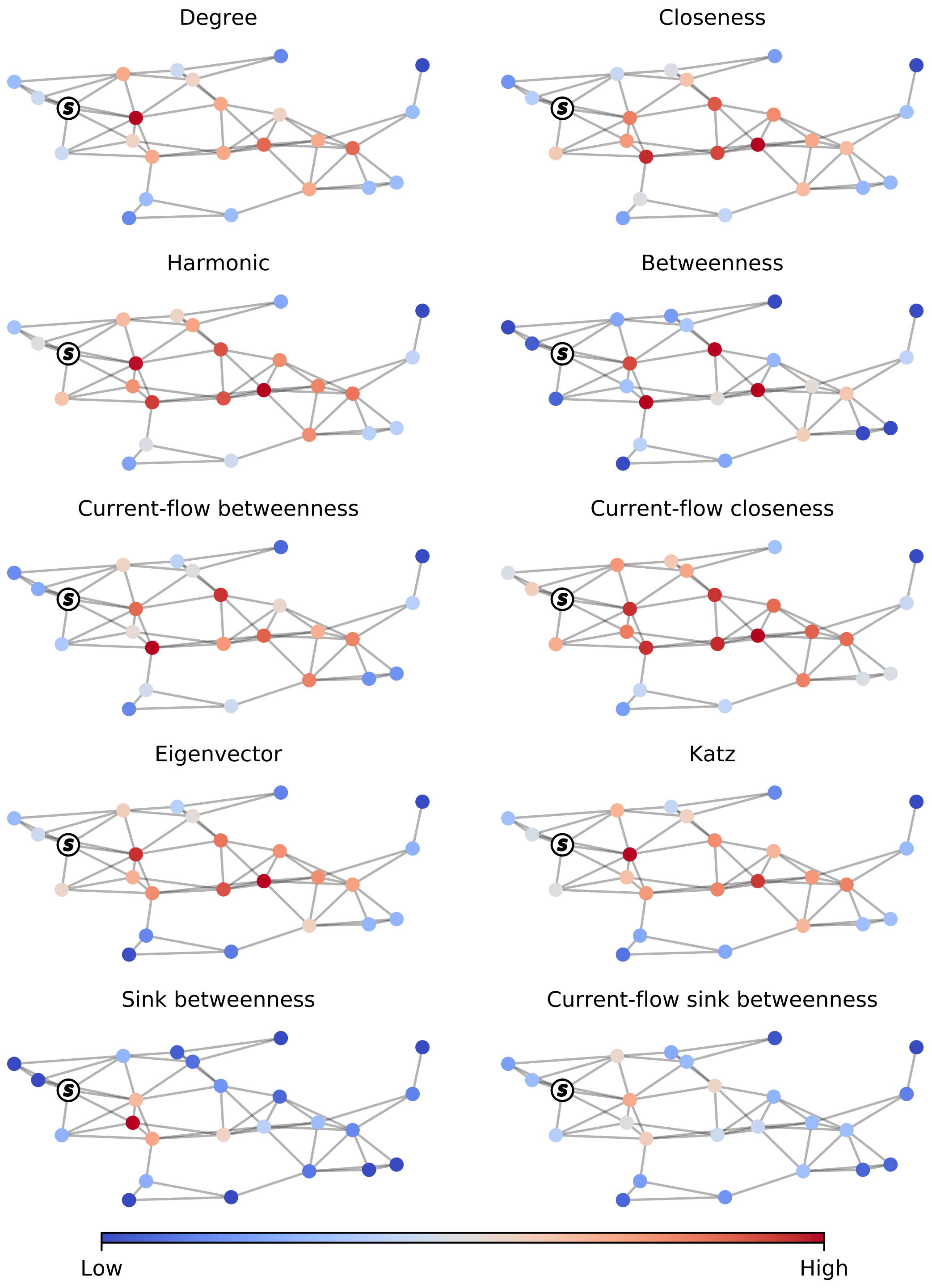


Figure 25: Node score value coloured according to different centrality measures.

5.3 EVALUATION

Since distinct centrality measures have different interpretations of what makes a node important, our objective is to analyse the performance of such metrics in ranking the impact that each node has on the connectivity of WSNs. However, there is no standard procedure to carry out an analysis of this kind. Therefore, we performed three different experiments, which we describe below in this section. Lastly, once we determine which centrality measure is best-suited for our needs, we instantiate our health model using the metric ranking as an approximation of nodes' importance.

5.3.1 *Settings*

In all experiments, we used the NetworkX [138] python package to generate the network graphs and compute the centrality measures scores. While centrality measures are usually evaluated over the Erdős-Rényi or Barabási-Albert models, in this work, we use Random Geometric Graphs (RGG) as they are a more realistic representation of WSNs [139, 140, 141]. Nodes are placed uniformly at random in the unit square, and any two nodes are connected by an edge if the distance between them is within a transmission radius r. Aside from requiring the network to be connected, we do not impose any other restrictions on the network structure, such as a minimum k-connectivity, presence or absence of cycles, or centralizing the sink node.

5.3.2 *Counting Paths*

When we examine the connectivity of a WSN, we are not particularly interested in the connection between any two arbitrary nodes. Instead, we are concerned about nodes being connected to the sink. Therefore, one way of measuring the network connectivity is by counting the total number of paths connecting the sink to each node in the network, let us name this value T_{P_S}. We can then temporarily disconnect each node, one at a time, to recompute the number of paths between the sink and the other nodes, which we denote by R_{P_S}. Finally, we can rank the nodes such that node i outranks node j if $T_{P_S} - R_{P_{S_i}} > T_{P_S} - R_{P_{S_j}}$ or simply $R_{P_{S_i}} < R_{P_{S_j}}$. Meaning, a node whose disconnection removes more paths is considered more important.

For this experiment, we assume the ranking obtained by counting paths as the ground truth and analyse the correlation with each ranking given by the centrality measures described in Section 5.2. We do this computation over 100 random geometric graphs of order 15. In particular, we compute the Kendall's τ and the Spearman ρ correlation coefficients. Both values range in the interval $[-1, 1]$ indicating strong disagreement around -1, no correlation around 0, and strong agreement around 1. Thus, the higher the correlation value, the better is the metric. Typically, Spearman correlation coefficients tend to be higher than Kendall's.

The mean correlation coefficients over the 100 graphs can be seen in Table 10. We can observe that the current-flow sink betweenness measure presents the highest correlation, followed by sink betweenness and closeness centrality, while Katz, eigenvector, and degree centrality present the lowest values. Also, except for the eigenvector centrality, the Spearman ρ is higher than Kendall's τ for all metrics. Which means that the eigenvector centrality is probably giving low ranks to some nodes that receive high ranks when counting paths or vice-versa.

There are two issues when counting all paths connecting the sink to network nodes. The first issue is that counting all these paths can be really expensive — which is the reason why this experiment is only performed over graphs with 15 nodes. While a single path can be found in $O(V + E)$ time, the number of paths in a network can be enormous, for instance, $O(n!)$ for a complete graph of order n. The second issue is that not all paths contribute to the sink connectivity. Let us consider the paths $\{\mathcal{S}, v_1, v_3\}$ and $\{\mathcal{S}, v_1, v_2, v_3\}$.

Table 10: Mean correlation coefficients between metrics and ranking by counting paths.

$\|V\|$	Centrality	Kendall τ	Spearman ρ
15	Degree	0.192659	0.209119
	Closeness	**0.313026**	**0.365837**
	Harmonic	0.259808	0.302857
	Betweenness	0.293757	0.360487
	Cf betweenness	0.286593	0.360374
	Cf closeness	0.276209	0.312523
	Eigenvector	0.190848	0.180138
	Katz	0.185320	0.194818
	Sink betweenness	**0.333496**	**0.390212**
	Cf sink betweenness	**0.481101**	**0.577486**

Node v_2 is not giving the sink an alternative independent path to v_3 since it still requires node v_1 which is already considered in the first path.

5.3.3 *Counting Node Independent Paths*

Any two paths in the network that connect the same two non-adjacent nodes are said to be *node-independent* if they do not have any internal nodes in common. According to Menger's theorem, the number of node-independent paths between two nodes is always equal to the minimum number of nodes that must be removed to disconnect them.

This experiment is the same as the previous one, except this time we rank a node n by counting the remaining number of independent paths between the sink and each other network node when n is temporarily disconnected. Table 11 shows the mean correlation coefficients over 100 graphs of order 50. It is possible to notice that, in comparison to the previous experiment, the correlation of some metrics increased while others decreased. Current-flow sink betweenness still has the highest correlation, but this time is followed by current-flow betweenness, betweenness, and sink betweenness, these three presenting very similar results. Katz, eigenvector, and degree centrality measures still exhibit the lowest correlations. This time the Spearman ρ is higher than Kendall's τ for all metrics.

Table 11: Mean correlation coefficients between metrics and ranking by counting node independent paths.

$\lvert V \rvert$	Centrality	Kendall τ	Spearman ρ
50	Degree	0.117484	0.147835
	Closeness	0.254139	0.327075
	Harmonic	0.220759	0.287228
	Betweenness	**0.408120**	**0.511300**
	Cf betweenness	**0.408739**	**0.516205**
	Cf closeness	0.227013	0.292216
	Eigenvector	0.122902	0.163820
	Katz	0.095954	0.124506
	Sink betweenness	**0.408180**	**0.496516**
	Cf sink betweenness	**0.516054**	**0.632177**

Table 12: Mean correlation coefficients between metrics and ranking by counting an approximation of node independent paths.

$\lvert V \rvert$	Centrality	Kendall τ	Spearman ρ
50	Degree	0.130927	0.165050
	Closeness	0.249960	0.321991
	Harmonic	0.226517	0.295999
	Betweenness	**0.390498**	**0.491875**
	Cf betweenness	**0.400498**	**0.506592**
	Cf closeness	0.221726	0.286106
	Eigenvector	0.138828	0.184423
	Katz	0.115615	0.151474
	Sink betweenness	**0.391404**	**0.477312**
	Cf sink betweenness	**0.511963**	**0.626929**
100	Degree	0.065708	0.089039
	Closeness	0.217286	0.290192
	Harmonic	0.187782	0.254746
	Betweenness	**0.340509**	**0.438525**
	Cf betweenness	**0.343290**	**0.441455**
	Cf closeness	0.198794	0.267816
	Eigenvector	0.089037	0.122594
	Katz	0.031533	0.049557
	Sink betweenness	**0.330092**	**0.416705**
	Cf sink betweenness	**0.457480**	**0.575776**

Although counting independent paths is less expensive than counting all paths this method still does not scale to large networks. We can improve its performance by using an approximation algorithm that gives a lower bound on the number of node-independent paths between two nodes [142]. Table 12 shows the results when using this algorithm over 100 graphs of order 50 and 100 graphs of order 100. The results on graphs of order 50 are very similar to the ones achieved without the approximation. While the mean correlation of every metric decrease on graphs of order 100, the analysis remains the same, thus indicating a similar behaviour on larger graphs.

5.3.4 *Disconnecting Nodes*

Another way of analysing the performance of different centrality measures is to simulate an attack that disconnects nodes sequentially (following the ranking provided by each metric) and observe the effects on the network [143, 144].

Once more we emphasize that, in the context of WSNs, we are interested in monitoring the number of nodes that remain connected to the sink. If the network gets fragmented into multiple components, only the nodes connected to the sink will continue to be functional since they will be the only ones capable of reporting their data and receiving new instructions. Accordingly, we assume an attack strategy that always disconnects the highest ranking node within the sink component until the sink is isolated. Figure 26 illustrates this process for a graph of order 50. Note that the goal is not to simply isolate the sink as fast as possible. If that was the case, one could merely disconnect all of the sink's neighbours and be done with it. There is a subtle difference. For instance, a node that is only connected to the sink and nothing else has no impact on the connectivity of the other nodes in the network. We are looking for a metric that is capable of identifying the most important nodes and, in the average case, this metric will be able to isolate the sink faster (requiring fewer disconnections) than others.

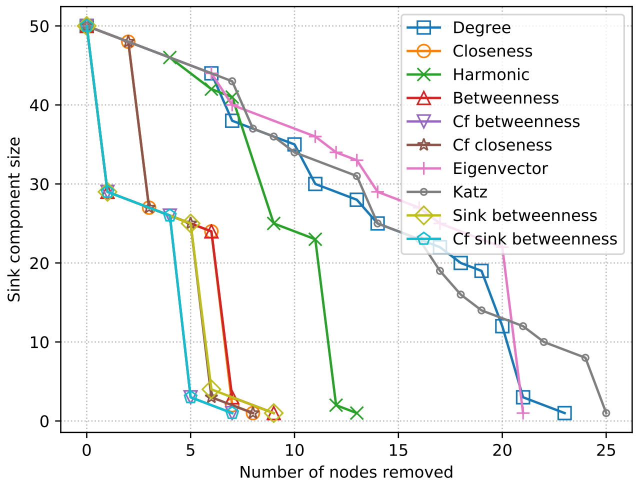


Figure 26: Sink component size as nodes are removed by highest score according to each centrality measure.

There are two possible disconnection strategies. The first one computes the centrality scores only once before removing any node. The second strategy recomputes the centrality scores for the remaining nodes of the sink component after each disconnection. We evaluate both strategies over graphs of different orders and the results are shown in Table 13 and Table 14. We compute the mean (μ) and standard deviation (σ) of the number of disconnected nodes and also rank the metrics among themselves (columns 1st, 2nd, ..., 10th) according to the number of nodes they need to disconnect to isolate the sink in each graph. Note that there can be ties, for instance, in the example illustrated in Figure 26, the rank would be: 1st current-flow sink betweenness and current-flow betweenness, 3rd current-flow closeness and closeness, 5th betweenness and sink betweenness, 7th harmonic, 8th eigenvector, 9th degree, and 10th Katz. The ranking fields of each row sum up to 100[1] while column sums might exceed 100 due to ties.

While the mean and standard deviation allow us to think about the average case, the ranking allows us to see that no metric is *always* better (fewer disconnections for sink isolation) than the others. However, we can observe that when using the first strategy,

1 Eigenvector and Katz centrality measures may fail to converge after a maximum number of iterations and are not ranked in such cases. We used the default parameters from NetworkX — 100 and 1000 maximum iterations respectively.

Table 13: Disconnecting nodes using the first strategy.

$\|V\|$	Centrality	μ	σ	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
50	Degree	17.4	5.0	3	2	8	7	10	12	8	27	17	6
	Closeness	15.5	4.9	8	4	5	5	16	21	17	14	9	1
	Harmonic	15.6	4.5	5	5	7	12	20	16	24	5	5	1
	Betweenness	14.0	4.6	2	3	25	21	7	10	14	4	7	7
	Cf betweenness	12.8	4.1	2	10	32	22	10	9	5	6	3	1
	Cf closeness	14.8	4.9	8	8	14	11	17	20	7	11	4	0
	Eigenvector	18.9	7.1	13	1	2	1	2	4	4	9	8	28
	Katz	17.7	5.6	9	4	10	3	4	5	5	18	29	12
	Sink betweenness	10.1	4.6	35	32	10	6	3	5	3	5	1	0
	Cf sink betweenness	**8.2**	**3.4**	**79**	**12**	**5**	**1**	**1**	**2**	**0**	**0**	**0**	**0**
100	Degree	33.9	8.0	0	3	4	5	7	10	6	32	25	8
	Closeness	28.2	10.8	1	7	8	6	20	17	12	10	16	3
	Harmonic	28.3	9.5	2	5	5	9	15	18	36	5	5	0
	Betweenness	23.6	8.5	0	3	16	33	10	10	10	8	10	0
	Cf betweenness	19.9	7.0	0	15	44	19	9	7	4	2	0	0
	Cf closeness	26.4	10.1	5	10	7	14	16	22	15	5	5	1
	Eigenvector	34.4	14.5	5	1	4	3	4	1	3	12	7	17
	Katz	33.7	10.6	2	4	4	5	4	8	7	19	23	6
	Sink betweenness	16.9	8.7	18	41	13	9	6	5	3	3	2	0
	Cf sink betweenness	**10.8**	**5.2**	**88**	**7**	**3**	**1**	**0**	**1**	**0**	**0**	**0**	**0**
500	Degree	204.8	54.8	0	1	2	1	22	14	16	38	6	0
	Closeness	203.2	72.5	0	3	3	3	8	11	22	43	7	0
	Harmonic	190.9	68.2	0	1	4	9	8	26	40	10	2	0
	Betweenness	134.0	47.5	0	0	7	56	15	12	7	2	1	0
	Cf betweenness	83.9	32.9	0	39	48	8	3	2	0	0	0	0
	Cf closeness	172.3	57.6	0	2	7	10	43	28	8	2	0	0
	Eigenvector	240.5	95.6	0	2	2	2	1	1	2	5	29	0
	Katz	107.0	0.0	0	0	0	1	0	0	0	0	0	0
	Sink betweenness	79.9	52.7	1	52	27	10	4	5	1	0	0	0
	Cf sink betweenness	**17.9**	**11.2**	**100**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Table 14: Disconnecting nodes using the second strategy.

$\|V\|$	Centrality	μ	σ	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
50	Degree	17.0	4.2	0	0	2	1	1	4	10	46	36	0
	Closeness	11.4	2.9	6	4	26	7	23	24	6	2	1	1
	Harmonic	13.9	3.3	0	2	6	2	3	14	66	7	0	0
	Betweenness	10.1	2.7	3	5	58	14	14	6	0	0	0	0
	Cf betweenness	9.9	2.7	3	5	63	15	13	0	1	0	0	0
	Cf closeness	10.9	2.4	6	6	34	8	30	14	1	1	0	0
	Eigenvector	10.0	0.0	0	0	0	0	0	1	0	0	0	0
	Katz	16.8	3.6	0	0	1	1	0	1	9	51	33	0
	Sink betweenness	**7.0**	**3.1**	**89**	**7**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**
	Cf sink betweenness	**7.0**	**3.0**	**85**	**12**	**2**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
100	Degree	33.7	7.0	0	0	0	0	0	0	4	65	31	0
	Closeness	18.1	4.6	0	2	9	8	29	47	5	0	0	0
	Harmonic	25.3	6.3	0	0	0	1	3	3	90	3	0	0
	Betweenness	14.3	4.0	1	2	50	35	11	1	0	0	0	0
	Cf betweenness	14.3	4.2	3	1	67	14	9	5	1	0	0	0
	Cf closeness	16.8	4.0	0	2	13	11	54	20	0	0	0	0
	Eigenvector	-	-	0	0	0	0	0	0	0	0	0	0
	Katz	32.9	7.4	0	0	0	0	0	0	1	38	39	0
	Sink betweenness	**8.8**	**3.8**	**87**	**13**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Cf sink betweenness	**9.1**	**4.0**	**73**	**25**	**2**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
500	Degree	197.3	31.9	0	0	0	0	0	0	4	96	0	0
	Closeness	70.1	17.2	0	0	0	0	65	35	0	0	0	0
	Harmonic	139.2	30.5	0	0	0	0	0	0	96	4	0	0
	Betweenness	40.1	11.3	0	0	33	67	0	0	0	0	0	0
	Cf betweenness	36.9	10.3	0	1	72	26	1	0	0	0	0	0
	Cf closeness	77.0	22.0	0	0	1	1	33	65	0	0	0	0
	Eigenvector	-	-	0	0	0	0	0	0	0	0	0	0
	Katz	170.0	11.0	0	0	0	0	0	0	0	0	2	0
	Sink betweenness	**14.3**	**6.7**	**62**	**38**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	Cf sink betweenness	**13.5**	**6.1**	**78**	**22**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

current-flow sink betweenness outperforms the other measures in most cases. In the second strategy, current-flow sink betweenness is slightly worse than sink betweenness for graphs of order 50 and 100, but is slightly better for graphs of order 500. We can also notice that when using the second strategy, fewer nodes have to be disconnected to isolate the sink. This approach is more time-consuming though, since the metric has to be recomputed several times. Therefore, choosing a particular strategy involves a trade-off between efficiency and computation power.

5.3.5 *Bringing it all together*

To instantiate the health model described in Section 5.1.2 we need to know the network topology, the importance of each node to the network operation, and which nodes are functional. Assuming we have the network topology, we can then compute the current-flow sink betweenness rank and use it as an approximation of nodes importance. We can use the results of the security mechanism in place (for instance, the attestation mechanism or a combination of attestation with measurements inspection as seen in the previous chapters) to determine whether a node is functional or compromised. With this information, we can determine which nodes have a safe path to the sink and finally compute the health value.

Figure 27 illustrates the health evaluation of a network over time. Compromised nodes are shown in red, while nodes that are functional but do not have a safe path to the sink are shown in grey. Note that the health values might be different from the percentage of nodes connected to the sink at each point in time. For instance, on the top right case, when only one node has been compromised the health evaluates to $\mathcal{H}(W) = 0.9$ while there is a total of 23 nodes out of 24 still connected to the sink ($23/24 = 0.96$). That is because by using the current-flow sink betweenness we can get a better approximation of the connectivity impact in terms of how well connected the remaining nodes are. Observe also that in this example we are assuming all nodes execute the same tasks or tasks of equivalent value. However, if that was not the case, then we could use an additional function $f(v_i)$ that determines the value of each node's tasks, and use that function in conjunction with the current-flow sink betweenness ranking adding their values to specify each node importance ($I(v_i) = f(v_i) + C_{CB}(v_i)$).

5.4 RELATED WORK

Centrality measures have been previously applied in WSNs to fulfil several tasks, such as routing [136, 145, 146], topology control [147, 148, 149, 150], access control [151], connectivity restoration [152], clustering [153], and target tracking [154] to cite a few. While some of these works even propose new centrality measures, none of them perform a comprehensive analysis comparing metrics as shown in here. Moreover, none of these works are concerned with measuring the network health and the damage caused by a compromised node.

Labatut and Ozgovde [155] illustrate the applicability of different topological measures for the analysis of WSNs through simulated experiments. However, they only cover one centrality measure, *betweenness*, and an adapted version of it, *sink-betweenness*. Jain and Reddy [156], on the other hand, give a general overview of how some centrality measures could be applied to WSNs. Nonetheless, their work has no analytical evaluation or actual experiments to further support and evidence their claims.

The work of Cartledge et al. [144] is closer to our notion of connectivity damage and network health. Nevertheless, the authors assume that nodes can still meet their responsibilities even if they are part of different disconnected components. This is not the case in WSNs since nodes that are disconnected from the sink will never be able to report their data or receive new commands. For the same reason, the well-known work from Albert, Jeong, and Barabási [143] which quantifies damage measuring the size of the largest connected component, the average size of the remaining components and the mean vertex-vertex distance is not directly applicable to WSNs.

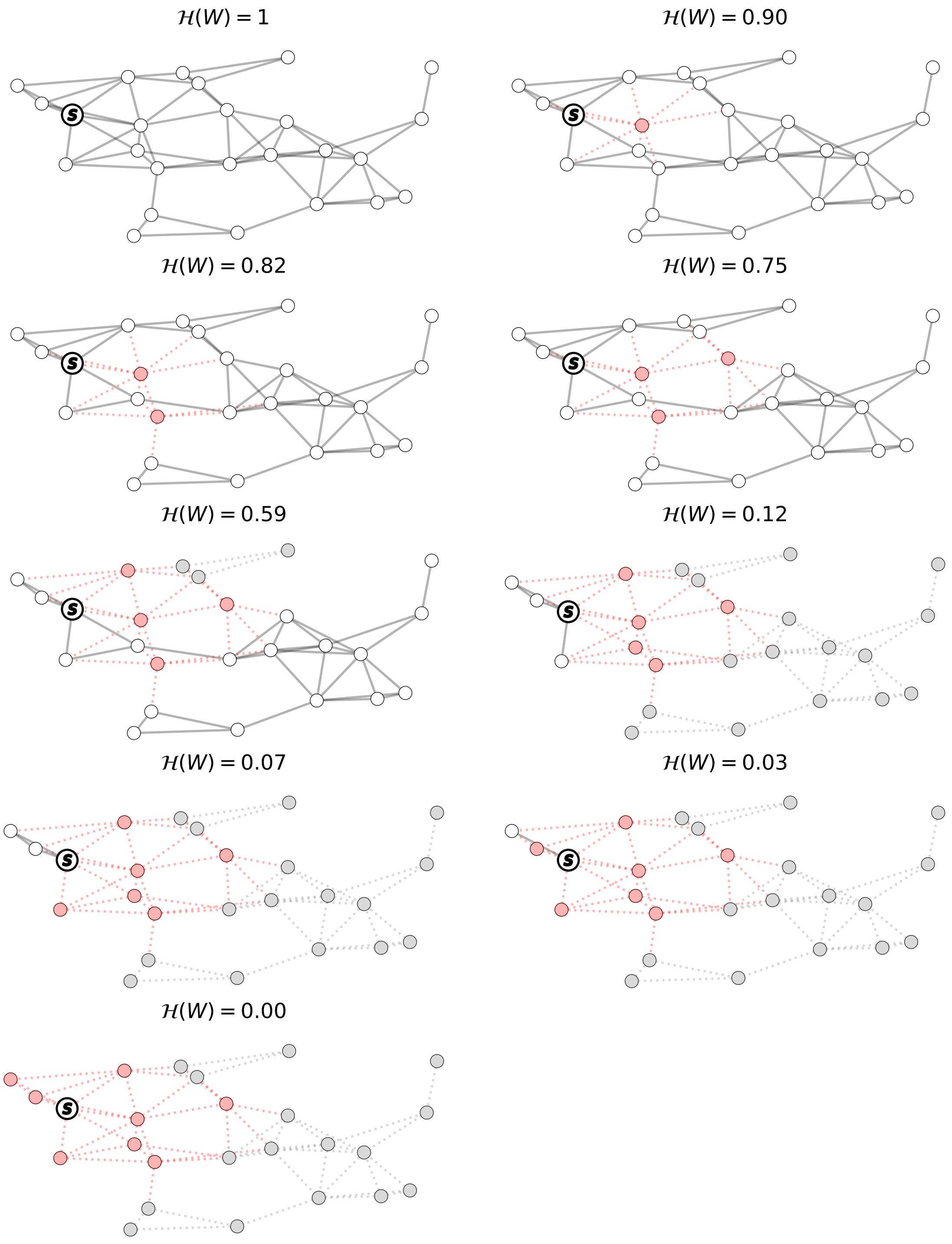


Figure 27: Health evaluation over time.

5.5 CONCLUSIONS

In this chapter, we have proposed a model capable of expressing the health of a WSN as a single value in the range $[0,1]$, where 0 means the network is not operational and 1 means it is working to its maximum capacity. A fundamental aspect to this model is the importance attributed to each node. In particular, we focused on how a node affects the network connectivity. To this end, we have performed an extensive analysis on how well centrality measures can rank the impact of each sensor node based on its topological position. We have conducted three different experiments which involved counting paths, independent paths, and disconnecting nodes from the network according to the metrics scores. Our results show that no metric is superior in all cases. However, we have proposed a novel metric named *current-flow sink betweenness* which is able to outperform existing metrics in most of the cases.

6 CONCLUSIONS

WSNs are becoming increasingly pervasive and will play a vital role in the IoT. Securing such systems is of utmost importance, and many defense solutions have been proposed to protect them from known attacks. However, the cybersecurity landscape is continuously evolving as new attacks emerge and, while prevention techniques are essential, they are not enough.

In this thesis, we have presented a framework to monitor the health and integrity of WSNs that allows us to comprehend to what degree a network can operate even in the presence of compromise. More specifically, this framework consists of security techniques to identify compromised devices and a mathematical model capable of expressing the network operational level.

Regarding security techniques, we focused on the use of attestation mechanisms to detect malicious devices. Not that other techniques could not be employed. So much so that we also proposed the use of attestation in combination with measurements inspection. However, the ability of attestation to detect, as long as the adversary modifies the software running on compromised devices, any type of attack (e.g., targeting confidentiality, integrity or availability) is what drew our attention, since it allows its use in many scenarios. While attestation is ineffective against physical attacks tampering a device's hardware, these attacks are difficult to scale. On top of that, most recent high-profile attacks, such as Stuxnet and the Mirai botnet, were performed exploiting software vulnerabilities.

As for the network operational level, we focused our evaluation on the impacts an attack might have on the network connectivity since it is crucial, in a WSN, for nodes to remain connected to the sink. Nevertheless, our proposed model to represent the health of a network can be easily extended to take into account other factors.

In the remainder of this chapter, we briefly summarize our main achievements and point out future research directions.

6.1 SUMMARY

In Chapter 2, we introduced a taxonomy that distinguishes the main characteristics of existing attestation mechanisms which allowed us to examine the tradeoffs between their design choices. We also identified four main topics that constitute open research issues: overoptimistic assumptions, effectiveness, time of check to time of use, and scalability; three of which we addressed in this thesis.

In Chapter 3, we proposed the combination of attestation with measurements inspection and designed three combination schemes: Detect and Attest, Group Subset Attestation, and Cascade. We evaluated all schemes both analytically and under simulations and showed that the D&A and GSA schemes offer a detection performance very close to attestation while consuming significantly less energy. While attestation provides accuracy close to 100% with a power consumption overhead of 33-58% and measurements inspection has an accuracy close to 50% with overhead close to 0, the combination schemes allows us to choose an accuracy in the range 96-99% with an overhead in the range 1-10%.

In Chapter 4, we reviewed a critical assumption made by software-based attestation mechanisms and proposed a novel stochastic approach. Instead of using the maximum known network RTT to send a single large attestation challenge, we take advantage of the fact that most of the time the actual network RTT is much smaller. Therefore, we use a series of short challenges in such a way that there is a high probability at least one will be replied in time. Our experimental results in real networks showed that when compared to the current state of the art solution our proposal reduces the overall attestation time and energy consumption around seven times for honest devices and two times for malicious ones, while improving the detection rate of honest devices (8% higher TPR) without compromising security (0% FPR).

In Chapter 5, we introduced a mathematical model to represent the health of WSNs that outputs a single value indicating its operational level. This model combines the knowledge regarding which nodes have been compromised with additional information that quantifies the importance of each node. In particular, we focused on the importance of each node for the network connectivity and investigated how well centrality measures can rank the nodes. In this process, we proposed a new metric named *current-flow sink betweenness*. Through a number of experiments, we show that no measure is invariably better in identifying sensors' connectivity relevance. However, our proposed metric outperforms existing measures in the vast majority of cases.

6.2 FUTURE RESEARCH DIRECTIONS

When designing our attestation and measurements inspection combination schemes, we assumed an adversary performing malicious data injection attacks by modifying the code running on compromised devices. It is possible to expand our threat model to consider further attacks. For instance, an adversary that physically tampers with environmental conditions, or that quietly compromises nodes until it controls the majority of them to only then start injecting data. Attestation would not detect the first type of attack, while measurements inspection would have a poor detection performance in the second. In such extreme cases each technique should be able to raise alarms on their own. Furthermore, we could bring even more techniques into the mix, such as misuse-based intrusion detection [157], and evaluate the benefits of the resulting schemes.

While intuitively our stochastic software-based attestation approach hinders the execution of collusion, impersonation, and proxy attacks it would be interesting to perform further experiments to analyse exactly to which extent it prevents this attacks. Also, the performance of our stochastic approach could be possibly improved with the use of machine learning techniques to estimate the channel conditions and compute the parameters for the attestation rounds.

Although we proposed a general model to represent the health of WSNs, we focused our evaluation on the impacts to the network connectivity. Moreover, we analysed connectivity from a topological perspective. However, an actual routing protocol might not be able to find certain paths, disconnecting nodes that could remain connected. Furthermore, it would be interesting to evaluate the relationship between the connectivity impact estimation and routing algorithms in terms of packet delivery ratio, end-to-end delay, and network lifetime. While we investigated the use of centrality measures to identify the significance of each node, other techniques, such as core graphs and core paths [158, 159], could possibly better estimate a node's connectivity impact. Additionally, it is possible to extend our current model to consider cases where nodes can have different levels of functionality rather than a boolean functional/not functional representation as well as incorporating confidentiality aspects that impact a network's health. Finally, we believe our health model can be used not only to guide decisions after an attack is detected but can also support pre-deployment assessment stages.

BIBLIOGRAPHY

[1] Luís M. Borges, Fernando J. Velez, and António S. Lebres. Survey on the Characterization and Classification of Wireless Sensor Network Applications. *IEEE Communications Surveys and Tutorials*, 16(4):1860–1890, 2014.

[2] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[3] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.

[4] Nicky Woolf. Ddos attack that disrupted internet was largest of its kind in history, experts say. *The Guardian [online] Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet*, 2016.

[5] Chris Graham. NHS cyber attack: Everything you need to know about 'biggest ransomware' offensive in history. *The Telegraph [online] Available: http://www.telegraph.co.uk/news/2017/05/13/nhs-cyber-attack-everything-need-know-biggest-ransomware-offensive*, 2017.

[6] Yun Zhou, Yuguang Fang, and Yanchao Zhang. Securing wireless sensor networks: a survey. *IEEE Communications Surveys and Tutorials*, 10(3):6–28, 2008.

[7] Javier Lopez, Rodrigo Roman, Isaac Agudo, and Carmen Fernandez-Gago. Trust Management Systems for Wireless Sensor Networks: Best Practices. *Computer Communications*, 33(9):1086–1093, 2010.

[8] Qijun Gu and Rizwan Noorani. Towards Self-propagate Mal-packets in Sensor Networks. In *Proceedings of the 1st ACM Conference on Wireless Network Security*, WiSec '08, pages 172–182, New York, NY, USA, 2008. ACM.

[9] Aleph One. Smashing The Stack For Fun And Profit. *Phrack Magazine*, 7(49), 1996.

[10] Hovav Shacham. The Geometry of Innocent Flesh on the Bone: Return-into-libc Without Function Calls (on the x86). In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 552–561, New York, NY, USA, 2007. ACM.

[11] Erik Buchanan, Ryan Roemer, Hovav Shacham, and Stefan Savage. When Good Instructions Go Bad: Generalizing Return-oriented Programming to RISC. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 27–38, New York, NY, USA, 2008. ACM.

[12] Aurélien Francillon and Claude Castelluccia. Code Injection Attacks on Harvard-architecture Devices. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 15–26, New York, NY, USA, 2008. ACM.

[13] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proceedings of the 13th Conference on USENIX Security Symposium*, SSYM '04, pages 223–238, Berkeley, CA, USA, 2004. USENIX Association.

[14] Rodrigo Vieira Steiner and Emil Lupu. Attestation in Wireless Sensor Networks: A Survey. *ACM Computing Surveys*, 49(3):51:1–51:31, 2016.

[15] Xinyu Jin, Pasd Putthapipat, Deng Pan, Niki Pissinou, and S. Kami Makki. Unpredictable Software-based Attestation Solution for Node Compromise Detection in Mobile WSN. In *Proceedings of the 29th IEEE Conference on Global Telecommunications Workshops*, GLOBECOM '10, pages 2059–2064, Piscataway, NJ, USA, 2010. IEEE Press.

[16] Benjamin Vetter and Dirk Westhoff. Simulation Study on Code Attestation with Compressed Instruction Code. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops*, PERCOM '12, pages 296–301, Washington, DC, USA, 2012. IEEE Computer Society.

[17] Shinsaku Kiyomoto and Yutaka Miyake. Lightweight Attestation Scheme for Wireless Sensor Network. *International Journal of Security and Its Applications*, 8(2):25–40, 2014.

[18] Hailun Tan, Wen Hu, and Sanjay Jha. A remote attestation protocol with Trusted Platform Modules (TPMs) in wireless sensor networks. *Security and Communication Networks*, 8(13):2171–2188, 2015.

[19] Xinyu Yang, Xiaofei He, Wei Yu, Jie Lin, Rui Li, Qingyu Yang, and Houbing Song. Towards a Low-Cost Remote Memory Attestation for the Smart Grid. *Sensors*, 15(8):20799–20824, 2015.

[20] N. Asokan, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, Matthias Schunter, Gene Tsudik, and Christian Wachsmann. SEDA: Scalable Embedded Device Attestation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 964–975, New York, NY, USA, 2015. ACM.

[21] Ahmad Ibrahim, Ahmad-Reza Sadeghi, Gene Tsudik, and Shaza Zeitouni. DARPA: Device Attestation Resilient to Physical Attacks. In *Proceedings of the 9th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '16, pages 171–182, New York, NY, USA, 2016. ACM.

[22] Hailun Tan, Gene Tsudik, and Sanjay Jha. MTRA: Multiple-tier remote attestation in IoT networks. In *IEEE Conference on Communications and Network Security*, CNS, pages 1–9. IEEE, 2017.

[23] Wei Feng, Yu Qin, Shijun Zhao, and Dengguo Feng. AAoT: Lightweight attestation and authentication of low-resource things in IoT and CPS. *Computer Networks*, 134:167–182, 2018.

[24] Claude Castelluccia, Aurélien Francillon, Daniele Perito, and Claudio Soriente. On the Difficulty of Software-based Attestation of Embedded Devices. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 400–409, New York, NY, USA, 2009. ACM.

[25] M van Steen. *Graph Theory and Complex Networks: An Introduction*. Maarten van Steen, April 2010.

[26] Aurélien Francillon, Quan Nguyen, Kasper B. Rasmussen, and Gene Tsudik. A Minimalist Approach to Remote Attestation. In *Proceedings of the 2014 Conference on Design, Automation and Test in Europe*, DATE '14, pages 1–6, 3001 Leuven, Belgium, 2014. European Design and Automation Association.

[27] Anupam Datta, Jason Franklin, Deepak Garg, and Dilsun Kaynar. A Logic of Secure Systems and its Application to Trusted Computing. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, SP '09, pages 221–236, Washington, DC, USA, 2009. IEEE Computer Society.

[28] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. Principles of remote attestation. *International Journal of Information Security*, 10(2):63–81, 2011.

[29] Frederik Armknecht, Ahmad-Reza Sadeghi, Steffen Schulz, and Christian Wachsmann. A Security Framework for the Analysis and Design of Software Attestation. In *Proceedings of the 20th ACM Conference on Computer and Communications Security*, CCS '13, pages 1–12, New York, NY, USA, 2013. ACM.

[30] Li Li, Hong Hu, Jun Sun, Yang Liu, and Jin Song Dong. Practical Analysis Framework for Software-Based Attestation Scheme. In *Proceedings of the 16th International*

Conference on Formal Engineering Methods, ICFEM '14, pages 284–299, Switzerland, 2014. Springer International Publishing.

[31] Diomidis Spinellis. Reflection As a Mechanism for Software Integrity Verification. *ACM Transactions on Information and System Security*, 3(1):51–62, 2000.

[32] Dazhi Zhang and Donggang Liu. DataGuard: Dynamic Data Attestation in Wireless Sensor Networks. In *Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '10, pages 261–270, Washington, DC, USA, 2010. IEEE Computer Society.

[33] Taejoon Park and Kang G. Shin. Soft Tamper-Proofing via Program Integrity Verification in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 4(3):297–309, 2005.

[34] Tamer AbuHmed, Nandinbold Nyamaa, and DaeHun Nyang. Software-Based Remote Code Attestation in Wireless Sensor Network. In *Proceedings of the 28th IEEE Conference on Global Telecommunications*, GLOBECOM '09, pages 1–8, Piscataway, NJ, USA, 2009. IEEE Press.

[35] Thanassis Giannetsos and Tassos Dimitriou. Spy-Sense: Spyware Tool for Executing Stealthy Exploits Against Sensor Networks. In *Proceedings of the 2Nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy*, HotWiSec '13, pages 7–12, New York, NY, USA, 2013. ACM.

[36] Glenn Wurster, Paul C. Van Oorschot, and Anil Somayaji. A Generic Attack on Checksumming-Based Software Tamper Resistance. In *Proceedings of the 26th IEEE Symposium on Security and Privacy*, SP '05, pages 127–138, Washington, DC, USA, 2005. IEEE Computer Society.

[37] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Pioneer: Verifying Code Integrity and Enforcing Untampered Code

Execution on Legacy Systems. *ACM SIGOPS Operating Systems Review*, 39(5):1–16, 2005.

[38] Yanlin Li, Yueqiang Cheng, Virgil Gligor, and Adrian Perrig. Establishing Software-Only Root of Trust on Embedded Systems: Facts and Fiction. In *Cambridge International Workshop on Security Protocols*, pages 50–68. Springer, 2015.

[39] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. Distributed Software-based Attestation for Node Compromise Detection in Sensor Networks. In *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, SRDS '07, pages 219–230, Washington, DC, USA, 2007. IEEE Computer Society.

[40] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The Sybil Attack in Sensor Networks: Analysis & Defenses. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, IPSN '04, pages 259–268, New York, NY, USA, 2004. ACM.

[41] Stephen Checkoway, Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, Hovav Shacham, and Marcel Winandy. Return-oriented Programming Without Returns. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 559–572, New York, NY, USA, 2010. ACM.

[42] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. SCUBA: Secure Code Update By Attestation in Sensor Networks. In *Proceedings of the 5th ACM Workshop on Wireless Security*, WiSe '06, pages 85–94, New York, NY, USA, 2006. ACM.

[43] Rick Kennell and Leah H. Jamieson. Establishing the Genuinity of Remote Computer Systems. In *Proceedings of the 12th Conference on USENIX Security Symposium*, SSYM '03, pages 295–310, Berkeley, CA, USA, 2003. USENIX Association.

[44] Arvind Seshadri, Adrian Perrig, Leendert Van Doorn, and Pradeep Khosla. SWATT: SoftWare-based ATTestation for Embedded Devices. In *Proceedings of the 25th IEEE*

Symposium on Security and Privacy, SP '04, pages 272–282, Washington, DC, USA, 2004. IEEE Computer Society.

[45] Jason Hill, Mike Horton, Ralph Kling, and Lakshman Krishnamurthy. The Platforms Enabling Wireless Sensor Networks. *Communications of the ACM*, 47(6):41–46, 2004.

[46] Mark Hempstead, Michael J. Lyons, David Brooks, and Gu-Yeon Wei. Survey of Hardware Systems for Wireless Sensor Networks. *Journal of Low Power Electronics*, 4(1):11–20, 2008.

[47] Michael Healy, Thomas Newe, and Elfed Lewis. Wireless Sensor Node Hardware: A Review. In *Proceedings of the 7th IEEE Sensors Conference*, pages 621–624, Washington, DC, USA, 2008. IEEE Computer Society.

[48] Ana Paula R. da Silva, Marcelo H. T. Martins, Bruno P. S. Rocha, Antonio A. F. Loureiro, Linnyer B. Ruiz, and Hao Chi Wong. Decentralized Intrusion Detection in Wireless Sensor Networks. In *Proceedings of the 1st ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks*, Q2SWinet '05, pages 16–23, New York, NY, USA, 2005. ACM.

[49] Rodrigo Roman, Jianying Zhou, and Javier Lopez. Applying Intrusion Detection Systems to Wireless Sensor Networks. In *Proceedings of the 3rd IEEE Consumer Communications and Networking Conference*, CCNC '06, pages 640–644, Piscataway, NJ, USA, 2006. IEEE Press.

[50] Bo Sun, Lawrence Osborne, Yang Xiao, and Sghaier Guizani. Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Communications*, 14(5):56–63, 2007.

[51] Trusted Computing Group TCG. *TPM Main Specification Level 2 Version 1.2, Revision 116*, 2011. Available at: `http://www.trustedcomputinggroup.org/resources/tpm_main_specification`, Last accessed: August 2018.

[52] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical One-Way Functions. *Science*, 297(5589):2026–2030, 2002.

[53] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160, New York, NY, USA, 2002. ACM.

[54] Daniele Perito and Gene Tsudik. Secure Code Update for Embedded Devices via Proofs of Secure Erasure. In *Proceedings of the 15th European Conference on Research in Computer Security*, ESORICS'10, pages 643–662, Berlin, Heidelberg, 2010. Springer-Verlag.

[55] Ross Anderson and Markus Kuhn. Tamper Resistance: A Cautionary Note. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, volume 2 of *WOEC '96*, pages 1–11, Berkeley, CA, USA, 1996. USENIX Association.

[56] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 104–113, Berlin, Heidelberg, 1996. Springer-Verlag.

[57] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, Berlin, Heidelberg, 1999. Springer-Verlag.

[58] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *Proceedings of the 3rd International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '01, pages 251–261, Berlin, Heidelberg, 2001. Springer-Verlag.

[59] Klaus Kursawe, Dries Schellekens, and Bart Preneel. Analyzing Trusted Platform Communication. In *Proceedings of the ECRYPT Workshop on Cryptographic Advances in Secure Hardware*, CRASH '05, 2005.

[60] Bernhard Kauer. OSLO: Improving the Security of Trusted Computing. In *Proceedings of the 16th USENIX Security Symposium*, SS '07, pages 1–9, Berkeley, CA, USA, 2007. USENIX Association.

[61] Evan R. Sparks. A Security Assessment of Trusted Platform Modules. Technical report, TR2007-597, Department of Computer Science, Dartmouth College, 2007.

[62] Bryan Parno. Bootstrapping Trust in a "Trusted" Platform. In *Proceedings of the 3rd Conference on Hot Topics in Security*, HOTSEC '08, pages 1–6, Berkeley, CA, USA, 2008. USENIX Association.

[63] Johannes Winter and Kurt Dietrich. A Hijacker's Guide to Communication Interfaces of the Trusted Platform Module. *Computers & Mathematics with Applications*, 65(5):748–761, 2013.

[64] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[65] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl. Side-channel Analysis of PUFs and Fuzzy Extractors. In *Proceedings of the 4th International Conference on Trust and Trustworthy Computing*, TRUST '11, pages 33–47, Berlin, Heidelberg, 2011. Springer-Verlag.

[66] Clemens Helfmeier, Christian Boit, Dmitry Nedospasov, and Jean-Pierre Seifert. Cloning Physically Unclonable Functions. In *IEEE International Symposium on Hardware-Oriented Security and Trust*, HOST '13, pages 1–6, 2013.

[67] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon, and Daniele Perito. SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. In *Proceedings of the 19th Network and Distributed System Security Symposium*, NDSS '12, pages 1–15. Internet Society, 2012.

[68] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. TrustLite: A Security Architecture for Tiny Embedded Devices. In *Proceedings of the 9th European Conference on Computer Systems*, EuroSys '14, pages 1–14, New York, NY, USA, 2014. ACM.

[69] Umesh Shankar, Monica Chew, and J. Doug Tygar. Side Effects Are Not Sufficient to Authenticate Software. In *Proceedings of the 13th Conference on USENIX Security Symposium*, SSYM '04, Berkeley, CA, USA, 2004. USENIX Association.

[70] Rick Kennell and Leah H. Jamieson. An analysis of proposed attacks against genuinity tests. Technical report, CERIAS, Purdue University, West Lafayette, 2004.

[71] Umesh Shankar, Monica Chew, and J. Doug Tygar. Side Effects Are Not Sufficient to Authenticate Software: Addendum. Technical report, EECS Department, University of California, Berkeley, 2004.

[72] Adrian Perrig and Leendert Van Doorn. Refutation of "On the Difficulty of Software-Based Attestation of Embedded Devices", 2010.

[73] Aurélien Francillon, Claude Castelluccia, Daniele Perito, and Claudio Soriente. Comments on "Refutation of On the Difficulty of Software-Based Attestation of Embedded Devices", 2010.

[74] Lodewijk F.W. van Hoesel, Tim Nieberg, Harm J. Kip, and Paul J.M. Havinga. Advantages of a TDMA based, energy-efficient, self-organizing MAC protocol for WSNs. In *Proceedings of the 59th IEEE Vehicular Technology Conference*, VTC '04, pages 1598–1602, 2004.

[75] Chongkyung Kil, Emre C. Sezer, Ahmed M. Azab, Peng Ning, and Xiaolan Zhang. Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, DSN '09, pages 115–124, Washington, DC, USA, 2009. IEEE Computer Society.

[76] Joonho Kong, Farinaz Koushanfar, Praveen K. Pendyala, Ahmad-Reza Sadeghi, and Christian Wachsmann. PUFatt: Embedded Platform Attestation Based on Novel Processor-Based PUFs. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, pages 1–6, New York, NY, USA, 2014. ACM.

[77] Young-Geun Choi, Jeonil Kang, and DaeHun Nyang. Proactive Code Verification Protocol in Wireless Sensor Network. In *Proceedings of the 2007 International Conference on Computational Science and Its Applications*, ICCSA '07, pages 1085–1096, Berlin, Heidelberg, 2007. Springer-Verlag.

[78] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[79] Mark Shaneck, Karthikeyan Mahadevan, Vishal Kher, and Yongdae Kim. Remote Software-based Attestation for Wireless Sensors. In *Proceedings of the 2nd European Conference on Security and Privacy in Ad-Hoc and Sensor Networks*, ESAS '05, pages 27–41, Berlin, Heidelberg, 2005. Springer-Verlag.

[80] Christian Collberg, Clark Thomborson, and Douglas Low. Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs. In *Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '98, pages 184–196, New York, NY, USA, 1998. ACM.

[81] Fred Douglis. The Compression Cache: Using On-line Compression to Extend Physical Memory. In *Proceedings of the 1993 USENIX Winter Conference*, pages 519–529, Berkeley, CA, USA, 1993. USENIX Association.

[82] Crispin Cowan, Calton Pu, Dave Maier, Heather Hintony, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-overflow Attacks. In *Proceedings of the 7th Conference on USENIX Security Symposium*, SSYM '98, pages 63–78, Berkeley, CA, USA, 1998. USENIX Association.

[83] Arvind Seshadri, Mark Luk, and Adrian Perrig. SAKE: Software Attestation for Key Establishment in Sensor Networks. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '08, pages 372–385, Berlin, Heidelberg, 2008. Springer-Verlag.

[84] Tigist Abera, N Asokan, Lucas Davi, Jan-Erik Ekberg, Thomas Nyman, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. C-FLAT: Control-FLow ATtestation for Embedded Systems Software. *arXiv preprint arXiv:1605.07763*, 2016.

[85] Eric Schkufza, Rahul Sharma, and Alex Aiken. Stochastic Superoptimization. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, pages 305–316, New York, NY, USA, 2013. ACM.

[86] Nuno P Lopes and José Monteiro. Weakest precondition synthesis for compiler optimizations. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 203–221. Springer, 2014.

[87] Nuno P. Lopes, David Menendez, Santosh Nagarakatte, and John Regehr. Provably Correct Peephole Optimizations with Alive. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '15, pages 22–32, New York, NY, USA, 2015. ACM.

[88] Ronald L. Rivest and Burt Kaliski. Rsa problem. In *Encyclopedia of cryptography and security*, pages 532–536. Springer, 2005.

[89] Oktay Ureten and Nur Serinken. Wireless security through rf fingerprinting. *Canadian Journal of Electrical and Computer Engineering*, 32(1):27–33, 2007.

[90] Saeed Ur Rehman, Kevin W. Sowerby, and Colin Coghill. Analysis of Impersonation Attacks on Systems Using RF Fingerprinting and Low-end Receivers. *Journal of Computer and System Sciences*, 80(3):591–601, 2014.

[91] David A. Knox and Thomas Kunz. Wireless Fingerprints Inside a Wireless Sensor Network. *ACM Transactions on Sensor Networks*, 11(2):37:1–37:30, 2015.

[92] Ferdinand Brasser, Kasper B Rasmussen, Ahmad-Reza Sadeghi, and Gene Tsudik. Remote Attestation for Low-End Embedded Devices: the Prover's Perspective. In *Design Automation Conference (DAC)*, 2016.

[93] Ing-Ray Chen, Yating Wang, and Ding-Chau Wang. Reliability of wireless sensors with code attestation for intrusion detection. *Information Processing Letters*, 110(17):778–786, 2010.

[94] Yi Yang, Sencun Zhu, and Guohong Cao. Improving Sensor Network Immunity Under Worm Attacks: A Software Diversity Approach. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '08, pages 149–158, New York, NY, USA, 2008. ACM.

[95] Sergey Bratus, Nihal D'Cunha, Evan Sparks, and Sean W. Smith. TOCTOU, Traps, and Trusted Computing. In *Proceedings of the 1st International Conference on Trusted Computing and Trust in Information Technologies: Trusted Computing - Challenges and Applications*, Trust '08, pages 14–32, Berlin, Heidelberg, 2008. Springer-Verlag.

[96] Xeno Kovah, Corey Kallenberg, Chris Weathers, Alexander Herzog, Matthew Albin, and John Butterworth. New Results for Timing-Based Attestation. In *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, SP '12, pages 239–253, Washington, DC, USA, 2012. IEEE Computer Society.

[97] Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti. Control-flow Integrity. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 340–353, New York, NY, USA, 2005. ACM.

[98] Christopher Ferguson and Qijun Gu. Self-Healing Control Flow Protection in Sensor Applications. *IEEE Transactions on Dependable and Secure Computing*, 8(4):602–616, 2011.

[99] Enes Goktas, Elias Athanasopoulos, Herbert Bos, and Georgios Portokalidis. Out of Control: Overcoming Control-Flow Integrity. In *Proceedings of the 35th IEEE Symposium on Security and Privacy*, SP '14, pages 575–589, Washington, DC, USA, 2014. IEEE Computer Society.

[100] Shuo Chen, Jun Xu, Emre C. Sezer, Prachi Gauriar, and Ravishankar K. Iyer. Non-control-data Attacks Are Realistic Threats. In *Proceedings of the 14th Conference on USENIX Security Symposium*, SSYM '05, Berkeley, CA, USA, 2005. USENIX Association.

[101] Miguel Castro, Manuel Costa, and Tim Harris. Securing Software by Enforcing Data-flow Integrity. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 147–160, Berkeley, CA, USA, 2006. USENIX Association.

[102] Hong Hu, Shweta Shinde, Sendroiu Adrian, Zheng Leong Chua, Prateek Saxena, and Zhenkai Liang. Data-oriented programming: On the expressiveness of non-control data attacks. In *Proceedings of the 37th IEEE Symposium on Security and Privacy*, SP '16, pages 969–986, Washington, DC, USA, 2016. IEEE Computer Society.

[103] Per Larsen, Andrei Homescu, Stefan Brunthaler, and Michael Franz. SoK: Automated Software Diversity. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 276–291, Washington, DC, USA, 2014. IEEE Computer Society.

[104] Michael Backes, Thorsten Holz, Benjamin Kollenda, Philipp Koppe, Stefan Nürnberger, and Jannik Pewny. You Can Run but You Can'T Read: Preventing Disclosure Exploits in Executable Code. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1342–1353, New York, NY, USA, 2014. ACM.

[105] Stephen Crane, Christopher Liebchen, Andrei Homescu, Lucas Davi, Per Larsen, Ahmad-Reza Sadeghi, Stefan Brunthaler, and Michael Franz. Return to Where? You Can't Exploit What You Can't Find. In *Proceedings of Black Hat USA*, 2015.

[106] Ahmad-Reza Sadeghi and Christian Stüble. Property-based Attestation for Computing Platforms: Caring About Properties, Not Mechanisms. In *Proceedings of the 2004 New Security Paradigms Workshop*, NSPW '04, pages 67–77, New York, NY, USA, 2004. ACM.

[107] Liqun Chen, Hans Löhr, Mark Manulis, and Ahmad-Reza Sadeghi. Property-Based Attestation Without a Trusted Third Party. In *Proceedings of the 11th International Conference on Information Security*, ISC '08, pages 31–46, Berlin, Heidelberg, 2008. Springer-Verlag.

[108] Aarthi Nagarajan, Vijay Varadharajan, Michael Hitchens, and Eimear Gallery. Property Based Attestation and Trusted Computing: Analysis and Challenges. In *Proceedings of the 3rd International Conference on Network and System Security*, NSS '09, pages 278–285, Washington, DC, USA, 2009. IEEE Computer Society.

[109] Vittorio P. Illiano and Emil C. Lupu. Detecting malicious data injections in wireless sensor networks: A survey. *ACM Computing Surveys*, 48(2):24:1–24:33, October 2015.

[110] V. P. Illiano, L. Munoz-Gonzalez, and E. C. Lupu. Don't fool me!: Detection, characterisation and diagnosis of spoofed and masked events in wireless sensor networks. *IEEE Transactions on Dependable and Secure Computing*, 14(3):279–293, May 2017.

[111] NICTA. Castalia simulator, 2007.

[112] Texas Instruments. Chipcon-cc2420. *TEXAS INSTRUMENTS,[Online]. Available: http://www. ti. com/product/cc2420.[Accessed 31 1 2015]*, 2013.

[113] Jose A. Gutierrez, Edgar H. Callaway, and Raymond Barrett. *IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks*. IEEE Standards Office, New York, NY, USA, 2003.

[114] Allaboutbatteries.com, Jan 2011.

[115] Moreno Ambrosin, Mauro Conti, Ahmad Ibrahim, Gregory Neven, Ahmad-Reza Sadeghi, and Matthias Schunter. Sana: Secure and scalable aggregate network attestation. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 731–742, New York, NY, USA, 2016. ACM.

[116] Xavier Carpent, Karim ElDefrawy, Norrathep Rattanavipanon, and Gene Tsudik. Lightweight swarm attestation: A tale of two lisa-s. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 86–100, New York, NY, USA, 2017. ACM.

[117] Sapon Tanachaiwiwat and Ahmed Helmy. Correlation Analysis for Alleviating Effects of Inserted Data in Wireless Sensor Networks. In *MobiQuitous*, pages 97–108. IEEE Computer Society, 2005.

[118] V. Chatzigiannakis and S. Papavassiliou. Diagnosing Anomalies and Identifying Faulty Nodes in Sensor Networks. *Sensors Journal, IEEE*, 7(5):637–645, May 2007.

[119] Mohsen Rezvani, Aleksandar Ignjatovic, Elisa Bertino, and Sanjay Jha. A robust iterative filtering technique for wireless sensor networks in the presence of malicious attacks. In Chiara Petrioli, Landon P. Cox, and Kamin Whitehouse, editors, *SenSys*, page 30. ACM, 2013.

[120] Intel Corporation. *Intel Edison Module Documentation*, 2015. Available at: `https://software.intel.com/en-us/iot/hardware/edison/documentation`, Last accessed: August 2018.

[121] Ryan Gardner, Sujata Garera, and Aviel D. Rubin. On the Difficulty of Validating Voting Machine Software with Software. In *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*, EVT'07, pages 11–11, Berkeley, CA, USA, 2007. USENIX Association.

[122] Qiang Yan, Jin Han, Yingjiu Li, Robert H. Deng, and Tieyan Li. A Software-based Root-of-trust Primitive on Multicore Platforms. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 334–343, New York, NY, USA, 2011. ACM.

[123] Xiaofei He, Xinyu Yang, Rui Li, and Qingyu Yang. A novel delay-resilient remote memory attestation for smart grid. In *Wireless Algorithms, Systems, and Applications*, pages 88–99. Springer, 2013.

[124] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. A topology discovery algorithm for sensor networks with applications to network management. In *IEEE CAS Workshop on Wireless Communications and Networking*, 2002.

[125] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. Stream: Sensor topology retrieval at multiple resolutions. *Telecommunication Systems*, 26(2-4):285–320, 2004.

[126] Guoqiang Mao, Barış Fidan, and Brian DO Anderson. Wireless sensor network localization techniques. *Computer networks*, 51(10):2529–2553, 2007.

[127] Christopher J Mallery, Sirisha Medidi, and Muralidhar Medidi. Relative localization with 2-hop neighborhood. In *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pages 1–4. IEEE, 2008.

[128] Yue Ivan Wu, Hao Wang, and Xiujuan Zheng. Wsn localization using rss in three-dimensional spacea geometric method with closed-form solution. *IEEE Sensors Journal*, 16(11):4397–4404, 2016.

[129] L C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[130] Y Rochat. Closeness centrality extended to unconnected graphs: The harmonic centrality index. In *ASNA*, 2009.

[131] M EJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[132] U Brandes and D Fleischer. Centrality measures based on current flow. In *STACS*, volume 3404, pages 533–544. Springer, 2005.

[133] K Stephenson and M Zelen. Rethinking centrality: Methods and examples. *Social networks*, 11(1):1–37, 1989.

[134] P Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[135] L Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[136] E MR Oliveira, H S Ramos, and A AF Loureiro. Centrality-based routing for wireless sensor networks. In *IFIP Wireless Days*, pages 1–5. IEEE, 2010.

[137] G Lawyer. Understanding the influence of all nodes in a network. *Scientific reports*, 5:8665, 2015.

[138] A Hagberg, P Swart, and D S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory, 2008.

[139] L Lima and J Barros. Random walks on sensor networks. In *5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks and Workshops*, pages 1–5. IEEE, 2007.

[140] H Kenniche and V Ravelomanana. Random geometric graphs as model of wireless sensor networks. In *Computer and Automation Engineering*, volume 4, pages 103–107. IEEE, 2010.

[141] H Zheng, F Yang, X Tian, X Gan, X Wang, and S Xiao. Data gathering with compressive sensing in wireless sensor networks: a random walk based approach. *IEEE Transactions on Parallel and Distributed Systems*, 26(1):35–44, 2015.

[142] D R White and M Newman. Fast approximation algorithms for finding node-independent paths in networks, 2001.

[143] R Albert, H Jeong, and AL Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[144] C L Cartledge and M L Nelson. Connectivity damage to a graph by the removal of an edge or a vertex. *arXiv preprint arXiv:1103.3075*, 2011.

[145] XH Li and ZH Guan. Energy-aware routing in wireless sensor networks using local betweenness centrality. *International Journal of Distributed Sensor Networks*, 9(5):307038, 2013.

[146] A Jain. Betweenness centrality based connectivity aware routing algorithm for prolonging network lifetime in wireless sensor networks. *Wireless Networks*, 22(5):1605–1624, 2016.

[147] A Cuzzocrea, A Papadimitriou, D Katsaros, and Y Manolopoulos. Edge betweenness centrality: A novel algorithm for qos-based topology control over wireless sensor networks. *Journal of Network and Computer Applications*, 35(4):1210–1217, 2012.

[148] L Sitanayah, K Brown, and C Sreenan. Fault-tolerant relay deployment based on length-constrained connectivity and rerouting centrality in wireless sensor networks. *Wireless Sensor Networks*, pages 115–130, 2012.

[149] H S Ramos, A Boukerche, A LC Oliveira, A C Frery, E MR Oliveira, and A AF Loureiro. On the deployment of large-scale wireless sensor networks considering the energy hole problem. *Computer Networks*, 110:154–167, 2016.

[150] X Fu, Y Yang, W Li, and G Fortino. Topology upgrading method for energy balance in scale-free wireless sensor networks. In *14th International Conference on Networking, Sensing and Control*, pages 192–197. IEEE, 2017.

[151] J Duan, D Gao, C Heng Foh, and H Zhang. Tc-bac: A trust and centrality degree based access control model in wireless sensor networks. *Ad Hoc Networks*, 11(8):2675–2692, 2013.

[152] I F Senturk and K Akkaya. Connectivity restoration in disjoint wireless sensor networks using centrality measures. In *39th Conference on Local Computer Networks Workshops*, pages 616–622. IEEE, 2014.

[153] A Jain and BV R Reddy. Eigenvector centrality based cluster size control in randomly deployed wireless sensor networks. *Expert Systems with Applications*, 42(5):2657–2669, 2015.

[154] N Meghanathan. An eigenvector centrality-based mobile target tracking algorithm for wireless sensor networks. *International Journal of Mobile Network Design and Innovation*, 6(4):202–211, 2016.

[155] V Labatut and A Ozgovde. Topological Measures for the Analysis of Wireless Sensor Networks. *Procedia Computer Science*, 10:397–404, 2012.

[156] A Jain and BVR Reddy. Node centrality in wireless sensor networks: Importance, applications and advances. In *IEEE 3rd International Advance Computing Conference (IACC)*, pages 127–131. IEEE, 2013.

[157] Ismail Butun, Salvatore D Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1):266–282, 2014.

[158] M Barrère, R V Steiner, R Mohsen, and E C Lupu. Tracking the Bad Guys: An Efficient Forensic Methodology To Trace Multi-step Attacks Using Core Attack Graphs. In *Proceedings of the 13th IEEE International Conference on Network and Service Management*, Nov 2017.

[159] M Barrère and E C Lupu. Naggen: a Network Attack Graph GENeration Tool. In *Proceedings of the IEEE Conference on Communications and Network Security*, Oct 2017.