Hiding the Topology of an IEEE 802.15.4 Network with Limited Energy

Sara Beatriz Schwarz*, Dimitrios-Georgios Akestoridis*, Patrick Tague*, and Hanan Hibshi*†
* *Carnegie Mellon University*
Email: sschwarz@andrew.cmu.edu, {akestoridis, tague, hhibshi}@cmu.edu
† *King Abdul-Aziz University*

***Abstract*—The Internet of Things is an ever-progressing area. As the number of Internet-connected devices grows, their security grows in complexity. The risk pertaining to the insecurity of these networks is also heightened by the fact that the user's physical environment can be affected by network attacks. The IEEE 802.15.4 wireless network standard provides MAC layer security but does not require MAC packet header encryption, which leaks useful information to an eavesdropper about the network topology. Prior work has demonstrated the use of obfuscation for wireless routing, but there is little existing work targeting energy-efficient topology obfuscation.**

Our work provides a method for obfuscating the network topology of an IEEE 802.15.4-based network by changing network association. Specifically, we propose the assignment of multiple aliases to each device during the network association phase to give the impression of many devices, wherein the role that a device presents can be further manipulated. We present two algorithms for coordinating aliases with the network authority, both providing multiple MAC addresses per device to degrade a potential attacker's ability to perform reconnaissance attacks. To evaluate our approach, we enhance the corresponding implementation in the popular ns-3 network simulator. We show that the achieved topology obfuscation comes in trade for increased overhead of the association process, both data overhead and latency, a trade-off that we study via simulation. Our ns-3 enhancements will be made publicly available as an open-source ns-3 branch, providing resources for the community and a way to validate and expand up on our work.

I. Introduction

The Internet of Things (IoT) and the concept of connecting our day-to-day objects is an ever progressing area and one that has been greatly commercialized. The proliferation of these systems also means a continued growth in their complexity. Wireless networks by nature have properties that make these systems harder to secure compared to wired devices. Introducing complexities to the equation, such as battery-based devices for example, may increase security threats; which motivates us to move away from traditional methods and having to think of new ideas for security.

Among the many security challenges brought on by the emergence of IoT, one of the most challenging to grasp is the fact that our assets are no longer digital, but rather expand into the physical world through sensing and actuation.

In their 2020 paper, Akestoridis et al. introduce an open source Zigbee network security analysis tool called Zigator [1]. Through this work they found that by pairing the short MAC addresses, stated by the IEEE 802.15.4 standard, between source and destination packets, an attacker can infer the network topology. This results primarily from the fact that the IEEE 802.15.4 standard provides for no security protections of MAC packet headers. Hence, we are motivated to address the question of how to hide or obfuscate this information. Though not a direct vulnerability, it can be leveraged as a stepping stone by attackers for building a greater threat and compromising the physical well-being of users.

Attack Model: The risk and harm of this leak is specific to a threat model that assumes an attacker to be a passive eavesdropper [2]: attackers that make use of information leaks at the reconnaissance phase to conduct a plan for a successful attack. In addition, the attacker will not hold any prior knowledge of the network topology since the main focus is acquiring that knowledge through the information leak. Finally, the attacker does not have the resources to do signal detection to infer the location of the devices or physically locate and count the number of devices in the network.

Project Motivation: As mentioned above, this project extends the work done in Zigator [1]. Zigbee [3] is one of the network protocols mostly used in commercial IoT and its lower layers are based on the IEEE 802.15.4 protocol. During their experimental setup, the authors found a class of attacks categorized as reconnaissance attacks [1] and provided a process for identifying the arrangement and roles of devices in a Zigbee network.

Figure 1 illustrates a Zigbee network and how the reconnaissance attack mentioned above looks like while considering our attack model using publicly available data from Akestoridis et al. [4]. The image shows a hub connected with another mains-powered device acting as a Zigbee router and a battery-based end device. Under this image we see a capture of packets, seen through the Wireshark [5]tool. We are able to see how the communication between this network looks like and by observing only the MAC header we can infer the topology as shown at the bottom of the image.

By leveraging this information, an attacker can passively detect the types of packets being sent in a network (e.g., Data Requests, which are sent by Zigbee End Devices and Routers), thus establishing a profile of device roles in the network. Since certain commands are issued only by devices in certain roles, this provides useful information for analyzing encrypted command traffic. The attacker could next launch a Denial of Service (DOS) attack to force a factory reset on a device [1]. The attacker could also forge and send beacon packets to all the Zigbee Routers, which would cause Personal Area Network Identifier (PAN ID) conflicts. The Zigbee Routers would contact the Zigbee Coordinator which would then set up a new PAN ID. If the attacker jams the network command that updates the PAN ID (which the attacker can detect by inspecting certain unencrypted header fields [1]) this could cause End Devices and Routers to not receive the command to update their PAN IDs, which can result in disconnections and is a further point of vulnerability in certain Zigbee networks.

Obfuscation is a method that looks to make the entity harder to comprehend. This contributes in exhausting the resources of malicious actors, changing their motivation or their attack method. No matter what, obfuscating the topology will add an extra step into the malicious actors' attack path. One commonly used obfuscation method in networks is using dummy packets. The dummy packets provide deceptive information to obfuscate whichever goal is considered.

In reality, security research is commonly aimed specifically at vulnerabilities that can be directly attacked, but most common attacks, like DDoS attacks, make use of reconnaissance attacks as the first phase in building the attack scenario [6]. One of the main problems for wireless networks has been the inability to provide a feasible protocol for securing communication at the MAC Layer. In decentralized networks, this is a bigger problem due to lack of structure and coordination in the system. Hence, the obtained non-encrypted information can make the hosts susceptible to active attacks [2].

As mentioned previously, the IEEE 802.15.4 protocol does not encrypt MAC headers. Though there is work done for adding encryption to the MAC header [7], but that work never mentions the energy constraint. In our work, we propose two algorithms that use obfuscation, specifically dummy packets, to fix this issue by giving an impression of many devices associating into the network. We provide an evaluation on the trade-off between obfuscation and energy consumption.

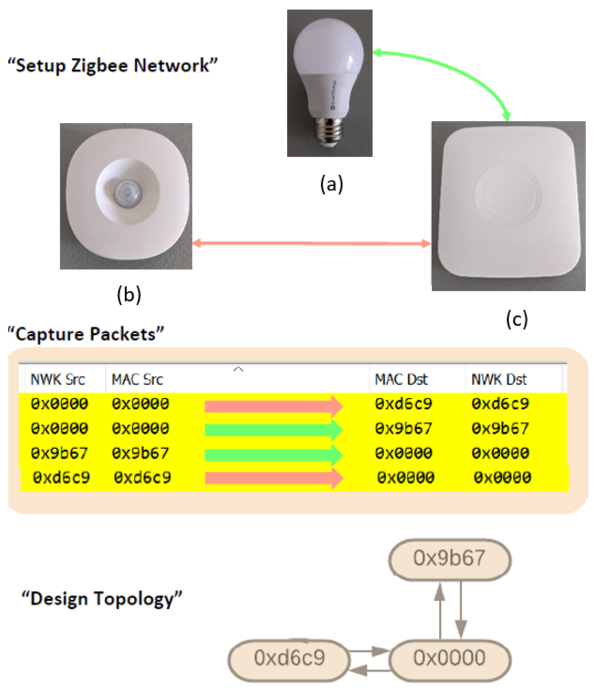
In the rest of this paper, we first cover background on the protocol and relevant concepts in Section II. We then give a summary of related approaches to fixing similar issues in Section III. In Section IV we present our obfuscation methods and their implementation in ns-3. Next, we provide the project's evaluation metrics in Section V-A. In Section V-B we go over our experimental setup and our simulated data (Section V-B1 to Section V-B2). Finally, we discuss potential future work and the contributions from this project in Section VI.

II. Background

Before diving into the details of our work, we provide some relevant information on wireless ad hoc networks, the IEEE 802.15.4 standard for low-power wireless networks, and the popular ns-3 network simulator.

A. Wireless Ad hoc Networks

Wireless networks consist of an infrastructure of devices or nodes that are connected through emission of electromagnetic



NWK Src	MAC Src		MAC Dst	NWK Dst
0x0000	0x0000	→	0xd6c9	0xd6c9
0x0000	0x0000	→	0x9b67	0x9b67
0x9b67	0x9b67	→	0x0000	0x0000
0xd6c9	0xd6c9	→	0x0000	0x0000

Fig. 1: In a Zigbee reconnaissance attack, an eavesdropper analyzes packet headers to identify devices and determine the network topology and roles of devices in the network. The devices shown are (a) a SmartThings Smart Bulb, (b) a SmartThings Motion Sensor (IM6001-MTP01), and (c) a SmartThings Hub (STH-ETH-200).

signals through the air rather than rather than wires [8]. These systems become harder to secure due to the openness of the medium, their additional technical challenges related to legacy wireless, complex physical layer implementations, and market pressures for affordability and flexibility [9].

Ad hoc networks lack a central infrastructure and use peer-to-peer communication, creating a space used commonly for fewer devices. In addition, ad hoc networks are constrained to low energy consumption to extend device life, since most of the devices used in these networks rely on battery power or other exhaustible means [10].

B. IEEE 802.15.4 standard

The IEEE 802.15.4 standard [11] is the foundation for the vast majority of protocols used for ad hoc networks (including Zigbee), due to its focus on low cost and low-power networking in commercial and residential environments. The standard provides the physical layer (PHY) and medium access control (MAC) sublayer specifications for low data-rate wireless connectivity among portable devices with limited battery consumption, targeting scenarios where tens to hundreds of kilobits-per-second of raw data throughput is sufficient.

1) Participating Devices: Two different device types can participate in an IEEE 802.15.4 network. First, a Full Function Device (FFD) is intended to serve as a coordinator or PAN coordinator, such as a Hub or a light bulb. Second, a Reduced Function Device (RFD) is intended for end devices, specifically devices that are extremely simple, such as a sensor;

2	1	variable	0/5/6/10/14	1	variable	2
Frame control	Sequence number	Address fields	Auxiliary security header	Command frame ID	Command payload	Frame check sequence
Frame header				MAC Payload		Footer

Fig. 2: MAC command frames are dedicated frame types in the IEEE 802.15.4 standard used to carry commands and their associated payload. The command frame structure is shown here, where the number above each field indicates the corresponding number of octets.

it does not have the need to send large amounts of data, and only associates with a single FFD at a time. Throughout this paper, a PAN coordinator is assumed to be an FFD, and an end device is assumed to be an RFD.

The IEEE 802.15.4 standard further differentiates between beacon enabled and non-beacon enabled networks, with the main difference being whether or not the PAN coordinator regularly emits beacons. Beacons are particularly useful in networks that require synchronization or heartbeat functionality. The PAN coordinators in non-beacon enabled networks wait for end devices to send beacon requests to initiate network interaction, such as transmitting all pending data for the end device after it wakes from low-power sleep. Another way of the networks' variability in energy maintenance is providing the RFDs, from the PAN Coordinator, with short MAC addresses to use after associating.

2) MAC Command Frame Format: The 2011 version standard defines four MAC frame structures: beacon, data, acknowledgement, and MAC command. These frames are used respectively for transmitting beacons, transferring data, confirming successful reception, and issuing commands. In Figure 2, one may observe that there is reserved space in the MAC command frame payload to set the command type identifier, which may, for example, identify a data or beacon request.

3) Transaction Process: One of the primary goals of the standard is to reduce energy consumption, which achieves it by using the MCPS-DATA.request and MLME-POLL.request primitives. Since an RFD will typically sleep for the majority of its lifetime, the PAN coordinator will queue all packets destined for an end device in an indirect transmission queue. Using the MLME-POLL.request, an RFD will be set to wake up at a certain frequency to verify for queued data from the coordinator by sending it an MCPS-DATA.request. These transaction control messages thus allow end devices to coordinate data exchanges with the PAN coordinator to make the best use of battery energy and sleep cycles.

4) Association Process: When an end device wants to join a PAN, it uses the association process defined in the standard and graphically outlined in Figure 3. The figure shows the MAC and network layer interaction between the coordinator (right side) and a device (left side). In a non-beacon enabled network, the device being associated, for example an RFD, initiates the process by scanning channels to find a network to join. Once identified, it sends a beacon request intended for

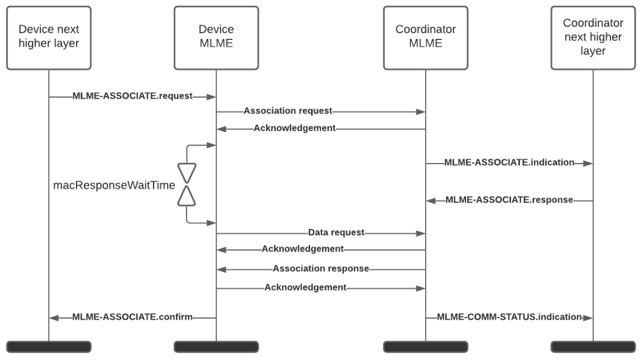


Fig. 3: The association process makes use of the data transaction method specific to this standard, and concludes with the FFD confirming the device's association by sending an association response containing the device's new short MAC address.

a coordinator. The coordinator replies with a beacon which contains information like the network PAN ID and the Coordinator address. Next, the device sends an association request to the coordinator. Once the coordinator receives the association request it responds with an acknowledgment which will set the end device into idle mode for a variable time to wait. Meanwhile, the coordinator processes the association request and proceeds to prepare an association response packet, with payload that contains the device's new short MAC address. This association response is added to the indirect transmission queue to wait for the device to poll for data. Once the end device wakes and receives the association response, it will confirm its association to the network and begin using its new short address.

5) Security Features: The IEEE 802.15.4 standard incorporates the AES-128 block cipher in CCM* mode for MAC layer encryption and authentication purposes [11]. Depending on what security level the protocol is set to, each frame may include a Message Integrity Code (MIC). However, the Zigbee standard suggests that higher-layer security is sufficient without this additional MAC layer protection [3].

C. ns-3: Network Simulator

ns-3 is a network simulator that uses the Discrete Event Simulation (DES) model, specifically used for research and education [12]. It is a free and completely community dependent project and used as the main simulating tool by many research entities. It has a feature of being able to provide packet capture (PCAP) files to visualize the packets in Wireshark [5].

The current ns-3 software includes a base skeleton of the IEEE 802.15.4 network model and associated protocol implementations. Unfortunately, the primitives specific to the standard's transaction method, association process, and energy consumption variability are incomplete or incorrect.

The current implementation of the ns-3 Low-Rate Wireless Personal Area Network (LR-WPAN) model is implemented under the assumption that all devices are never idle and are already associated into the network.

We identify other work that uses the ns-3 simulator and acknowledged and attempted to fix the lack of energy considerations in the LR-WPAN model, but we were unable to locate the implementation code for these works [13], [14]. We did not find work that addresses the lack of device association.

III. Related Work

While we are unaware of any prior work that designs topology obfuscation mechanisms specific for low-power wireless ad hoc networks, we provide a broader summary of relevant related work in the area of obfuscating network information against passive eavesdroppers. There is a range of work in this space, but there is a fair amount of difference in the actual asset that is being obfuscated or the means for obfuscation.

A. Obfuscating Network Traffic

Hayajneh et al. [15] propose two methods for obfuscating routing algorithms specifically for ad hoc networks. Both of their methods consider the energy constraint of these type of networks while providing techniques for anonymizing the source/destination of network traffic. They compare their techniques with related algorithms used for obfuscation using probabilistic packet forwarding. An attacker is assumed to already have knowledge of the network topology. Cao et al. [16] provide a physical solution for obfuscating network traffic. They build a framework which uses MIMO in order to do header blinding, a process of utilizing signal streams to interfere with real network data streams. Chaddad et al. [17] propose an algorithm to obfuscate the packet sizes leaked by network traffic, by looking at disclosed information from visible features that cannot be hidden through encryption, including packet sizes.

B. Using Dummy Packets for Obfuscation

Diyanat et al. [18] present the need for obfuscating transmitted data by using dummy packets to augment the data and preserve user privacy. They state the traffic rate of source nodes is useful in gaining more information about the participants. Knowing how augmentation requires extra resources, they provide a method which minimizes the augmented dummy packets to maintain their aim of reaching the "source rate privacy" by not affecting the original packets' delay distribution. Yang et al. [19] present a similar approach using dummy packets with the goal of hiding source locations. They correlate source location privacy to the occurrence of sensed events and demonstrate anonymity against a global adversary.

C. Network Topology Obfuscation

Contrary to the work mentioned above, the following describes a few deception tools used for obfuscating specifically the network topologies, though we were unable to find related work aimed at ad hoc networks. Meier et al. [20] created a tool called NetHide that does dynamic topology obfuscation by targeting path tracing probes. They demonstrate how their tool can obfuscate topologies without quality reduction. ProTO [21] is another system that adopts the detect-then-obfuscate framework. Similar to NetHide, this system looks to obfuscate network topologies by dynamically detecting probing behaviors and deceive the attackers by providing fake network information leading to topologies that are structurally accurate. These systems were not built targeting low power area networks, thus the energy consumption of utilizing a dynamic detection system while at the same time running a machine language algorithm for distinguishing probe behaviors would result in exceeding the devices' energy limits.

D. Header Encryption

Dalal et al. [7] propose the use of MAC header encryption to prevent information leakage in low-power wireless networking protocols. Unfortunately, this work does not give any consideration to energy consumption in battery-based devices or several other potential impacts of header encryption.

IV. Device Aliases for Topology Obfuscation

Now we present our approach to topology obfuscation in IEEE 802.15.4 networks. Our approach relies on the use of multiple *device aliases* to give the impression of a network topology that is far more elaborate than the real topology.

A. Assigning Aliases to Devices

We propose two approaches for allocating device aliases to achieve topology obfuscation: *MAC Layer Determined Aliases (MLDA)* and *Network Layer Determined Aliases (NeLDA)*. Both of these algorithms rely on using an aliasing method where each device has multiple MAC addresses that it can use to interact with the network. The aim is for each end device to have a list of Extended MAC addresses that it will use to associate to the network and use for continued communication, while appearing to be multiple different devices. To accomplish this, we must first assume that the network layer will route all aliased traffic from end devices to the intended coordinator, performing no filtering of aliases or aggregation of contents across groups of aliases.

The algorithms differ in the methods for acquiring the aliases, but both yield the desired result that the network appears to have a different topology than in reality. The design idea for relying on the association process is due to the assumption that eavesdroppers will have access to the traffic even when a device is first joining the network. Thus, as we will show, both implementations make sure to associate the fake addresses as soon as the device joins the network. This ensures that obfuscation is maintained for all devices joining the network after an attacker begins eavesdropping.

In both algorithms, each device i will have access to a pool of addresses, and it will choose to use a_i of these available addresses as aliases, in addition to its "real" MAC address, where $1 \leq a_i \leq a_{\max}$ and $a_{\max}$ is the size of the MAC address pool available to each device. The choice of a_i introduces a trade-off between identification of individual devices and their roles in the network and the amount of overhead required for network association. As a_i increases, an attacker will still be able to infer the topology, though it will be a larger topology that cannot differentiate real devices from aliases, effectively limiting the information leaked to the attacker for blueprinting a greater threat.

1) MAC Layer Determined Aliases (MLDA): Our MLDA algorithm relies on the existence of multiple Extended MAC addresses held by each end device, and it chooses its aliases from those available MAC addresses. Prior to association, each device is responsible for randomly generating its own a_i value, according to a predetermined probability distribution and an onboard random number generator (RNG). The device will then associate using each of a selection of a_i of its a_{max} available aliases and its real Extended MAC address, each separated by a random waiting time. The waiting time is needed to avoid bursts of associations that are obviously initiated by a single device. The MLDA algorithm relies on an extra primitive that once all aliases are associated, the device will privately reveal to the PAN coordinator the set of aliases that should correspond to the real device. In order for MLDA to achieve the desired obfuscation, we rely on an assumption that Network layer encryption is used for all post-association communication using an appropriate network key.

2) Network Layer Determined Aliases (NeLDA): Our NeLDA algorithm takes an alternate approach by having the PAN coordinator choose and distribute the MAC addresses that each end device should use as aliases. This eliminates the needs for end devices to hold pre-loaded lists of MAC addresses and to perform random number generation. In NeLDA, a new device will send an association request with a meaningless, random identifier to the PAN coordinator. The PAN coordinator will respond to this request by randomly choosing the number of MAC addresses a_i, choosing a_i MAC addresses to send to the device, and packaging them into a message with $(a_{\text{max}} - a_i)$ all-zero addresses to pad the message to a fixed length. This message is then sent in an encrypted association response to the device, for example using optional 802.15.4 MAC layer security mechanisms. The device will then re-associate to the PAN coordinator with each of the a_i received aliases and its true Extended MAC address as before. In the case of NeLDA, the PAN coordinator has network-wide control of how many and which aliases are assigned to end devices. In addition, since the coordinator is assigning aliases to the device, it does not require the additional step of identifying which aliases belong to which real device, as required by MLDA.

B. Implementation of Topology Obfuscation in ns-3

Although ns-3 already has an implementation of the LR-WPAN model, it is insufficient for our needs due to absence of the association process, data request, MAC command frame format, and the indirect transmission queue. As such, we put significant effort into implementing all of these capabilities in a private branch of ns-3 to be able to validate and evaluate our approach. We added functions for the MLME-SCAN.request, MLME-SCAN.confirm, MLME-ASSOCIATE.request, MLME-ASSOCIATE.response, MLME-ASSOCIATE.confirm, MLME-ASSOCIATE.indication primitives. We added the packet structure for the Beacon Request Command and an indirect transmission queue class. We also updated all functions that make use of the indirect transmission queue, the function for the MCPS-DATA.request primitive, and the data transaction functions called Pd Data Confirm and Pd Data Indication. All these changes were made in the MAC Class and MAC Payload Header class found in `mac.cc/.h` and `mac_plHeaders.cc/.h` respectively. In addition, since our work additionally targets the energy consumption and other overhead, we further expanded the ns-3 model to include simulation of idle mode and polling for data, as it is important to observe and compare the costs of our approaches. This effort has yielded a fully implemented primitive to test and evaluate the exchange of aliases between the PAN coordinator and end devices. As a secondary contribution of this work, we will soon release our expanded LR-WPAN implementation as an open-source branch of ns-3, found in the ns-3 source code at `src/lr-wpan/model/`.

V. Evaluation of Obfuscation Methods

In order to validate the contribution of our work, we need to measure how both algorithms, MLDA and NeLDA, are able to achieve topology obfuscation and the energy consumption that the obfuscation implementation adds to the standard approach.

A. Evaluation Metrics

We focus on two primary metrics to understand the trade-off between topology obfuscation and the associated cost.

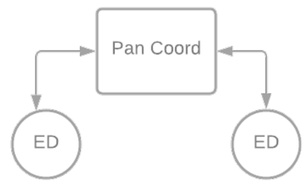
In order to capture the strength of the achieved topology obfuscation, we define the *visibility* as the ratio of total MAC addresses used to the number of devices, or, equivalently, the average number of aliases per device. With k devices, the visibility v_k is given by

$$v_k = \frac{1}{k} \sum_{i=1}^{k} (a_i + 1) \,. \tag{1}$$

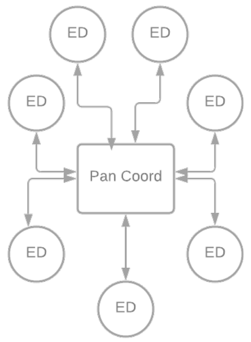
Intuitively, a higher visibility indicates a greater degree of uncertainty in the number of devices present, roughly corresponding to the extent to which the behavior of a single device is partitioned over multiple unique identities. Visibility can also be interpreted as a measure of the degree of difficulty the attacker would have in reconstructing the true network topology.

In addition to the visibility alone, the amount of randomness in each a_i parameter is an important consideration. Fully characterizing the behavior of this parameter, however, depends on making a specific assumption about its distribution, which we are not making. However, once known, this randomness could be captured using common statistical metrics such as variance, entropy, or others, relative to the assumed distribution.

To capture the cost of topology obfuscation, we measure the *aggregate data flow* as the total number of bytes transmitted and received by each end device during the association process, as a function of the number of aliases and which algorithm is used.



(a)



(b)

Fig. 4: Our simulated scenarios include a PAN coordinator (labeled "Pan Coord") with two and seven end devices (labeled "ED"), respectively, in (a) Scenario 1 and (b) Scenario 2.

With MLDA, the association process will involve associating under the true identity and the a_i aliases, yielding an aggregate data flow of

$$f_{k,\text{MLDA}} = d_{\text{MLDA}} \sum_{i=1}^{k} (a_i + 1) \,, \qquad (2)$$

where d_{MLDA} is the number of bytes sent and received in a single association process under MLDA. Based on the simulation, we state that a standard Zigbee association transaction, limiting it only to association request packets (21 bytes), data request packets sent by unassociated devices (18 bytes) and association response packets (27 bytes), involves transmission of 66 bytes. The MLDA primitive introduces an additional 17 bytes of overhead, so $d_{\text{MLDA}} = 83$ bytes in this case.

With NeLDA, the association process involves both associating under all $a_i + 1$ identities as well as exchanging the padded list of MAC addresses from the PAN coordinator. Since each extended MAC address is eight bytes in length, the aggregate data flow for NeLDA is given by

$$f_{k,\text{NeLDA}} = (8(a_{\max} + 1) + d_{\text{NeLDA}}) \sum_{i=1}^{k} (a_i + 1) \,, \qquad (3)$$

where d_{NeLDA} is the number of bytes sent and received in a single association process under NeLDA. Since NeLDA does not add overhead per association, the data per association is equal to that of a standard Zigbee association transaction at $d_{\text{NeLDA}} = 66$ bytes.

In comparing the communication overhead of MLDA and NeLDA from (2) and (3), we see that the cost of using MLDA depends primarily on the average number of aliases used in the deployment, whereas the cost of using NeLDA depends on both the average and the maximum number of aliases. As such, the two algorithms offer different trade-offs under different probability distributions. Our results in the next section further reflect this difference.

In addition to the data overhead associated with topology obfuscation, it is important to note that associating with multiple identities with random inter-association timing could add significant latency to the overall association process. The two main factors that contribute to this latency are the number of aliases and the inter-association timing, meaning the random timing between subsequent associations by the same device using different aliases. To capture this additional timing overhead, we define the *association latency* as the total duration of the association process including all aliases and waiting times. This latency now has two sources of randomness, one in the number of aliases a_i and one in the inter-association timing. We let $\tau_{i,j}$, for $1 \leq j \leq a_i$, denote the amount of time that device i waits between associating with its $(j-1)^{\text{th}}$ and j^{th} aliases, where we slightly abuse notation and let the real identity be the 0^{th} alias. The total association latency ℓ_i experienced by device i is then given by

$$\ell_i = a_i d / r_b + \sum_{j=1}^{a_i} \tau_{i,j} \,, \qquad (4)$$

where d is the number of bytes transmitted in the association (either d_{MLDA} or d_{NeLDA} and r_b is the transmission rate of the channel, so d/r_b is the transmission latency per association request.

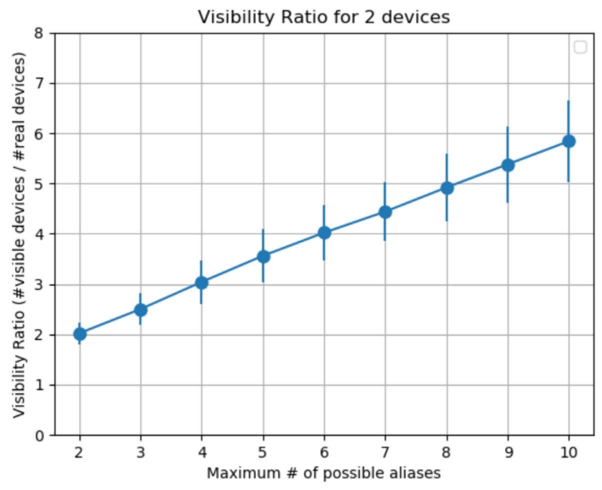
Here again, the association latency varies primarily with the average number of aliases employed by the devices. In addition, however, the actual inter-association timing is an important factor in fully understanding the cost of topology obfuscation. If the $\tau_{i,j}$ values are too small, then the associations from a single device will appear as a quick burst, which will present no challenge to the attacker in grouping them together per device. If the $\tau_{i,j}$ values are large, then the associations of multiple devices will mix together and be very difficult for the attacker to disambiguate, but the association process will be very time-consuming. When $\tau_{i,j}$ grows to provide stronger obfuscation, the resulting latency ℓ_i for MLDA and NeLDA will be very similar.

B. Simulation Study

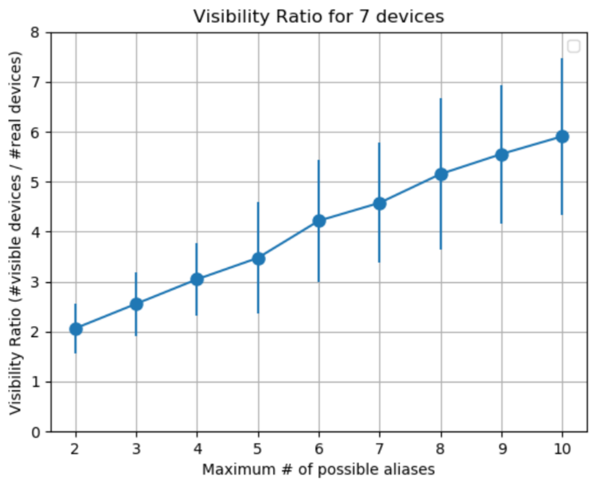
Using our ns-3 implementations, we set up our test environment with two different IEEE 802.15.4 network topologies. The first scenario, illustrated in Figure 4(a), has one PAN coordinator (FFD) connected to two end devices (RFDs). The second scenario, illustrated in Figure 4(b), has one PAN coordinator (FFD) connected to seven end devices (RFDs).

We ran the MLDA and NeLDA algorithms in the two scenarios illustrated in Figure 4 in a non-beacon enabled network, increasing the maximum number of aliases $a_{\max}$ in each subsequent trial, stopping at $a_{\max} = 10$. We ran each trial 25 times to account for random variations, and our results illustrate averages over these trials with error bars indicating standard deviation around the average.

Per the ns-3 documentation [12], the built-in L'Ecuyer's MRG32k3a random number generator (RNG) creates multiple uncorrelated substreams of output random numbers, and using different substreams leads to stronger resulting randomization.

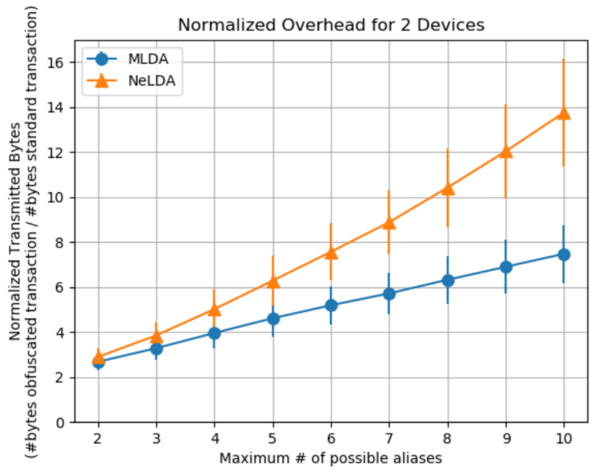


(a) Scenario 1

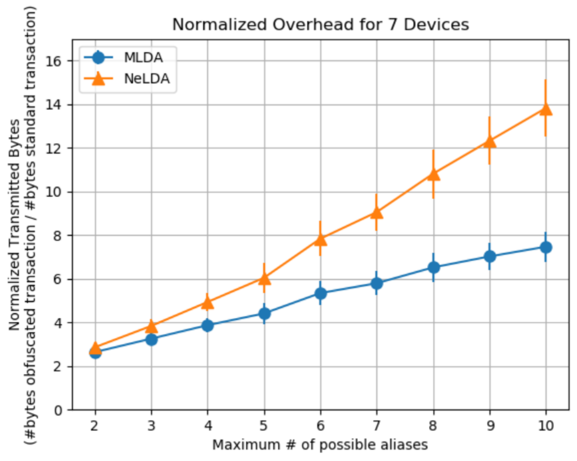


(b) Scenario 2

Fig. 5: The simulated visibility ratio is illustrated for the two scenarios illustrated in Figure 4 with a different number of end devices as a function of the maximum number of aliases $a_{\max}$.



(a) Scenario 1



(b) Scenario 2

Fig. 6: The simulated aggregate data flow is illustrated for the two scenarios illustrated in Figure 4 with a different number of end devices as a function of the maximum number of aliases $a_{\max}$.

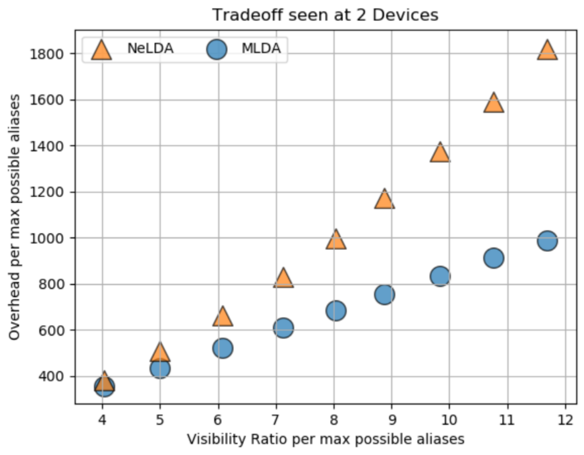
As such, we use a different RNG substream for each run of our simulation, rather than relying on different RNG seeds.

In our implementation, we chose all a_i values uniformly between 1 and $a_{\max}$.

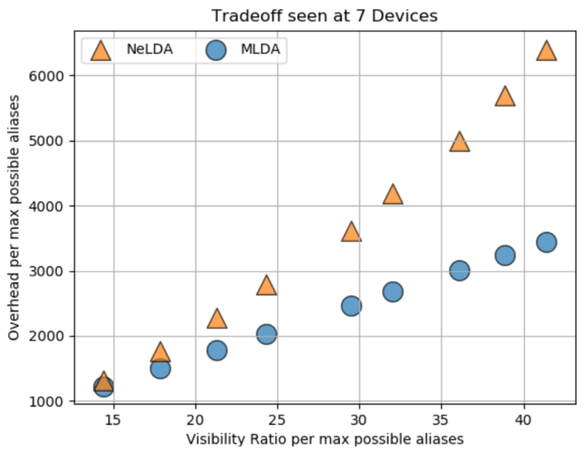
1) Visibility: The plots in Figure 5 illustrate the visibility ratio for Scenarios 1 and 2, respectively.

As expected, the visibility increases from 1, which represents no obfuscation, as the number of possible aliases increases. Due to the increased randomness, the error bars also grow as the number of possible aliases increases. Since scenario 2 involves more devices, the probability of having more varied number of aliases per device is increased. As mentioned in the evaluation section, this increases the obfuscation. Thus, the larger the standard deviation the more obfuscation in regards to attacker being able or not to find the number of aliases per device.

2) Data Overhead: Figure 6 represents the aggregate data flow overhead of both algorithms for Scenario 1 and Scenario 2. For ease of comparison, we normalize the data to the value with no obfuscation (66 bytes) and plot the increase in overhead instead of the raw value. As indicated by (2) and (3), both algorithms yield an increase in data overhead, but NeLDA increases at a faster rate due to dependence on both the average and maximum number of aliases, where MLDA increases linearly in the maximum (or average) number of aliases. As with the average data overhead, we also see that the standard deviation of the data overhead increases more rapidly with NeLDA than with MLDA. This is also due to



(a) Scenario 1



(b) Scenario 2

Fig. 7: Simulated visibility and data overhead results are plotted against each other to illustrate the trade-offs that can be achieved by varying the maximum number of aliases $a_{\max}$. The best-case scenario would land in the lower-right corner of the plot.

the compounded dependencies of aliases and padding to the maximum number of aliases exchanged.

We note that acknowledgement packets are not included in these plotted results for clarity of presentation, but they would have roughly similar impact on all results. It is also worth noting that the relatively lower standard deviations in Scenario 2 compared to Scenario 1 are due to the same values for data overhead being repeated for a larger number of devices, which yields lower variation in data overhead overall.

3) Trade-off: Figure 7 shows the trade-off between obfuscation and overhead in our methods for Scenario 1 and Scenario 2. The ideal goal is to reach values that fall in the bottom right. In other words, we are looking for something which has as much obfuscation as possible and the least amount of overhead increase as possible. As observed, between both algorithms and for both scenarios, MLDA is the closest to reaching this goal, considering NeLDA's super-linear growth. However, there are practical considerations at hand that do not unilaterally favor MLDA, which we discuss in Section VI.

C. Overhead Variation over Time

As the vast majority of the overhead of our obfuscation methods occur during the association phase, it is also worth demonstrating the relative overhead at other time of network operation. To do this, we measure the data overhead at different times after the start of the simulation, noting that the association occurs immediately, followed by normal network operation after the devices have joined the network. Figure 8 illustrates an example scenario with one end device with ten aliases, plotting data overhead in bytes as a function of time in seconds. The figure shows the overhead of the device of both obfuscation methods compared to unobfuscated operation both during association and during normal network operation. The idea is to represent the difference in overhead between the association process and the periodic data request polling packets. As previously observed, NeLDA has almost twice as much overhead increase during the association process as MLDA. However, as soon as the association process ends, the overhead falls back to additional 12 bytes per alias, where 12 refers to the size of the data request packets after the devices have been associated. We note that the differences between algorithms and the overall behavior of the example is influenced both by the number of aliases, which affects the vertical scaling and spacing between curves, and the randomization of inter-association times, which affects the horizontal position of the transition from association to normal operation (and back, if associations repeat occasionally).

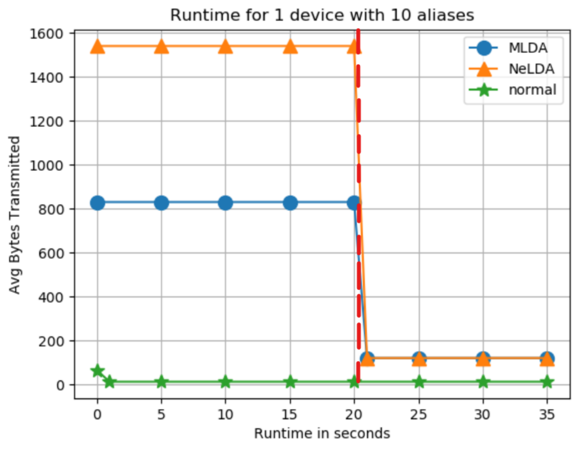


Fig. 8: At the left of the red dotted line is the overhead cost for the association process for both implementations. To the right of the line is the overhead cost during the rest of the device life, where all devices associate their aliases consecutively.

VI. Discussion and Conclusion

One metric that is not implemented nor measured in this paper is the randomness in the time interval between the association processes of the aliases. The time between each alias being associated needs to be random, as to avoid an attacker being able to designate packets as aliases based on a pattern. An implementation of this would only expand the time in which each fake address is associated, thus it would not affect the total overhead. In regards to obfuscation, as the distribution of the inter-association times $\tau_{i,j}$ skew higher, the strength of obfuscation will increase on average. Formal modeling and analysis of this timing behavior and the improvement in strength of the resulting obfuscation due to interleaving of associations is left as future work.

Another aspect to consider in this work for the algorithms' evaluation is the environment factors. Though both implementations are tested in topologies with one coordinator, a multihop environment won't require any change at an implementation level. The algorithms are largely dependent of the PAN coordinator, thus that ensures correct behavior. In addition, the environment can affect the choice between one algorithm and another going above the trade-off. Based on the results, MLDA is closer to the ideal trade-off compared to NeLDA, due to being half its overhead. Though this is ideal reach, there are limitations that make MLDA unusable. If the network is unstable MLDA may cause problems and increase the latency due to devices being disconnected. Thus, MLDA is more suitable for reliable networks with a large number of participating battery-based devices. In another case, NeLDA may be preferable, especially when considering networks with few battery-based devices.

In summary, we have demonstrated that the use of multiple aliases per device in an IEEE 802.15.4 network can effectively prevent an eavesdropper from inferring the true topology of a wireless network, including the identities and roles of the associated devices. We provide the MLDA and NeLDA algorithms for assigning aliases during network association, and we analyze the value and the overhead of the resulting topology obfuscation. To validate our designs, we enhance the IEEE 802.15.4 network model in ns-3 to provide a more realistic simulation and evaluation. Through our ns-3 simulation study, we show how MLDA and NeLDA differ in associated cost and how they effectively trade overhead for obfuscation strength. We will release our enhanced ns-3 implementation as an open-source branch to accompany this work.

References

[1] D.-G. Akestoridis, M. Harishankar, M. Weber, and P. Tague, "Zigator: Analyzing the security of zigbee-enabled smart homes," in *Proceedings of the 13th ACM Conf. on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '20. ACM, 2020, pp. 77–88. [Online]. Available: https://doi.org/10.1145/3395351.3399363

[2] J.-C. Kao and R. Marculescu, "Eavesdropping minimization via transmission power control in ad-hoc wireless networks," in *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, vol. 2, 2006, pp. 707–714. [Online]. Available: http://doi.org/10.1109/SAHCN.2006.288535

[3] *Zigbee Specification*, ZigBee Document 05-3474-21, ZigBee Alliance, 2015.

[4] D.-G. Akestoridis, M. Harishankar, M. Weber, and P. Tague, "CRAWDAD dataset cmu/zigbee-smarthome (v. 2020-05-26)," 2020. [Online]. Available: https://doi.org/10.15783/c7-nvc6-4q28

[5] The Wireshark team. (1998) Wireshark. [Online]. Available: https://www.wireshark.org/

[6] B. Harris, E. Konikoff, and P. Petersen, "Breaking the DDoS attack chain," *Institute for Software Research*, 2013.

[7] H. N. Dalal, N. V. Soni, and A. Razaque, "Header encryption of ieee 802.15.4," in *2016 IEEE LISAT Conf*, 2016, pp. 1–6. [Online]. Available: http://doi.org/10.1109/LISAT.2016.7494140

[8] P. Nicopolitidis, M. S. Obaidat, G. I. Papadimitriou, A. S. Pomportsis, *Introduction to Wireless Networks*. John Wiley & Sons, Ltd, 2002, ch. 1, pp. 1–24. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/047085801X.ch1

[9] Y. Xiao, H. Chen, S. Yang, Y.-B. Lin, and D.-Z. Du, "Wireless network security," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, no. 1, p. 532434, Dec 2009. [Online]. Available: https://doi.org/10.1155/2009/532434

[10] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *Proceedings of the 4th Annual ACM/IEEE International Conf. on Mobile Computing and Networking*, ser. MobiCom '98. ACM, 1998, pp. 181–190. [Online]. Available: https://doi.org/10.1145/288235.288286

[11] "IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (lr-wpans)," *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, 2011. [Online]. Available: http://doi.org/10.1109/IEEESTD.2011.6012487

[12] NSNAM. (2011) ns-3 network simulator. [Online]. Available: https://www.nsnam.org/about/what-is-ns-3/

[13] V. Rege and T. Pecorella, "A realistic MAC and energy model for 802.15.4," in *Proceedings of the Workshop on ns-3*, ser. WNS3 '16. New York, NY, USA: ACM, 2016, pp. 79–84. [Online]. Available: https://doi.org/10.1145/2915371.2915379

[14] J. M. Jose, A. Sikora, M. Schappacher, and N. M. Phuong, "Integration and analysis of an extended ieee 802.15.4 modules with SmartMAC and wake-on-radio functions into the network simulator NS3," in *2016 3rd IDAACS-SWS*, 2016, pp. 19–23. [Online]. Available: http://doi.org/10.1109/IDAACS-SWS.2016.7805778

[15] T. Hayajneh, R. Doomun, P. Krishnamurthy, and D. Tipper, "Source—destination obfuscation in wireless ad hoc networks," *Security and Communication Networks*, vol. 4, 08 2011. [Online]. Available: http://doi.org/10.1002/sec.220

[16] Y. Cao, A. Atya, S. Singh, Z. Qian, S. V. Krishnamurthy, T. Porta, P. Krishnamurthy, and L. Marvel, "Packet header obfuscation using MIMO," *IEEE/ACM Transactions on Networking*, vol. 28, no. 04, pp. 1712–1725, jul 2020. [Online]. Available: http://doi.org/10.1109/TNET.2020.2998398

[17] L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "Network obfuscation for net worth security," in *2020 Seventh International Conference on Software Defined Systems (SDS)*, 2020, pp. 83–88. [Online]. Available: http://doi.org/10.1109/SDS49854.2020.9143919

[18] A. Diyanat, A. Khonsari, and S. P. Shariatpanahi, "A dummy-based approach for preserving source rate privacy," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1321–1332, 2016. [Online]. Available: http://doi.org/10.1109/TIFS.2016.2515050

[19] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards event source unobservability with minimum network traffic in sensor networks," in *Proceedings of the First ACM Conf. on Wireless Network Security*, ser. WiSec '08. New York, NY, USA: ACM, 2008, pp. 77—88. [Online]. Available: https://doi.org/10.1145/1352533.1352547

[20] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: Secure and practical network topology obfuscation," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 693–709. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/meier

[21] T. Hou, Z. Qu, T. Wang, Z. Lu, and Y. Liu, "ProTO: Proactive topology obfuscation against adversarial network topology inference," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1598–1607. [Online]. Available: http://doi.org/10.1109/INFOCOM41043.2020.9155255