SAFE-IoT: Attesting Firmware in IoT Swarms using Volatile Memory and a Mixture of Experts

Varun Kohli
Electrical and Computer Engineering
National University of Singapore
Singapore
varun.kohli@u.nus.edu

Muhammad Naveed Aman
School of Computing
University of Nebraska-Lincoln
United States of America
naveed.aman@unl.edu

Biplab Sikdar
Electrical and Computer Engineering
National University of Singapore
Singapore
bsikdar@nus.edu.sg

***Abstract*—Advances in 5G mobile networks and artificial intelligence have led to rapid growth in the Internet of Things (IoT) as part of various smart initiatives. Embedded IoT microcontrollers are an easy target of firmware and network attacks, which become the root cause of various node-level and device-to-device (D2D) propagated anomalies. Thus, firmware integrity is essential to ensuring IoT security. Although several existing techniques require a legitimate copy of the device's firmware, authentic firmware may not be available. In addition, the available literature also has limitations in terms of scalability, computational complexity, low availability, and the need for specialized hardware. This paper presents Swarm Attestation of Firmware in Embedded-IoT (*SAFE-IoT*) to solve these problems using a Mixture of Denoising Autoencoder Networks (MoDAE) framework and is the first to use Static Random Access Memory (SRAM) to attest firmware in IoT swarms. We present a volatile memory dataset for swarm attestation, which contains thirteen network scenarios that capture various D2D relationships. *SAFE-IoT* achieves a 99+% attestation rate on authorized firmware, a 100% detection rate on anomalous firmware, and a 95+% detection rate on D2D propagated anomalies. The proposed method has a latency of 10^{-4} seconds per node. Lastly, we analyze robustness against perturbation of SRAM traces.**

***Index Terms*—Internet of Things (IoT), Swarm Attestation, Mixture of Experts (MoE), Static Random Access Memory (SRAM), Machine Learning, Anomaly Detection**

I. Introduction

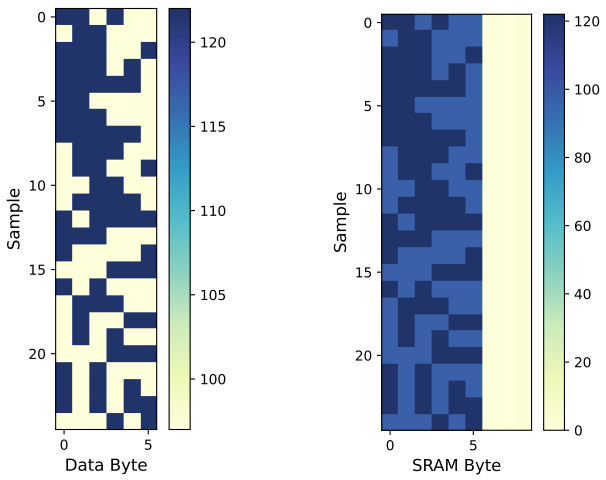
The present decade has seen rapid development in the Internet of Things (IoT), and billions of devices are deployed worldwide across initiatives for smart factories, homes, grids, intelligent transportation, healthcare, and defense owing to development in 5G communication, edge computing, and artificial intelligence technologies [1]. Embedded devices such as microcontrollers are a significant portion of these networks. They are usually found in swarm settings, a typical example of which is the intra-vehicle Controller Area Networks (CAN) deployed in automobiles, which comprise up to seventy microcontrollers performing different tasks ranging from sensing to control [2].

A recent study revealed that over 95% of vulnerabilities in IoT devices were related to their firmware [3]. Furthermore, abnormal behavior originating at one node in a swarm may have downstream effects due to device-device (D2D) propagation of information in the swarm. Thus, ensuring the authenticity of firmware on IoT devices is essential to the security of IoT networks, and various firmware attestation techniques have been proposed in the literature. Traditional methods involve the computation of a hash over the prover's flash memory over multiple iterations [4–6]. However, such approaches require a copy of the firmware, which may not be available due to intellectual property (IP) rights. They also have high computational complexity and latency, which is undesirable in real-time scenarios. Various hardware-based attestation approaches have been proposed to solve the complexity problem, which, however, require specialized hardware that may not be available on the IoT devices [7–9]. Hybrid attestation techniques such as [10, 11] cause low availability in IoT devices. Various swarm attestation methods have also been proposed [12–14], which have relatively high latency and involve complex cryptographic computations on the devices.

A study from 2022 introduced Static Random Access Memory (SRAM) as a potential feature for firmware attestation at the node level [15]. The SRAM has a significantly smaller size, by several orders of magnitude, than the flash memory dump and contains run-time information of the firmware running on the device. Inspired by their work, this paper proposes an SRAM-based swarm attestation scheme that achieves scalability, low latency, and high availability of the microcontrollers while overcoming the IP problem. It highlights the ability of the SRAM to help detect source and downstream anomalies originating from anomalous firmware in different parts of the swarm. To the best of our knowledge, this is the first study on volatile memory-based attestation of IoT swarms. The contributions of this article are as follows:

1) A novel attestation approach, *SAFE-IoT*, is proposed using a combination of fully connected and convolutional Denoising Autoencoders Experts (DAEs) or-

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research Development Programme, under grant FCP-NUS-RG-2022-019. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.



(a) Six bytes of random binary data (ASCII of 'a' or 'z', x-axis) over twenty-five samples generated at an arbitrary node, N_i.

(b) Received data embedded in the integer equivalent of hexadecimal content in the SRAM (x-axis) at node, N_{i+1}.

Fig. 1: Relationship between data shared by a sender and the SRAM contents of the receiver.

ganized as a Mixture of Experts (MoE). This work provides a proof of concept for using the devices' SRAM contents for the reliable attestation of IoT swarms.

2) A dataset[1] for SRAM-based swarm attestation is presented. It encompasses thirteen network scenarios of a four-node IoT swarm with two normal, two physical twins, and nine anomalous scenarios. It may be used for firmware attestation using anomaly detection, classification, and time-series tasks. The linked drive folder contains the dataset and preliminary analysis presented in Section V.
3) Detection results are shown for nine anomalous scenarios and two physical twin cases to show attestation and repeatability on production networks. In addition, experimental analyses on latency and robustness against byte-incremental noise are also presented.
4) A thorough literature review is conducted, and *SAFE-IoT* is compared with relevant past works regarding latency and detection.

The remainder of this article is organized as follows: Section II discusses related works on single-node and swarm attestation in IoT networks. The network and threat model considered in this study is introduced in Section III. The proposed method, *SAFE-IoT*, is discussed in Section IV followed by details of the experimental setup, *SAFE-IoT* dataset, and some preliminary analysis presented in Section V. Section VI presents the experimental results, and the study is concluded in Section VII.

[1]https://drive.google.com/drive/folders/1R5XSLNPmd3PTGzZfMvjC7NmZk_Crq9dh?usp=sharing

II. Related Works

In this section, we present some of the related works on firmware attestation. We look at three overall research types: software-based, hardware-based, and hybrid firmware attestation. Lastly, we will look at some swarm attestation techniques and highlight the benefits of *SAFE-IoT* over the discussed literature.

Notable studies on software-based attestation include SWATT [4], SCUBA [5], SAKE [6], [16] and [15]. To compute a collective hash, [4] requires nearly fifty thousand iterations to detect anomalies, while [5, 6] use a similar technique for malicious sensor nodes. These studies require complicated computation on the devices, have high latency, and are difficult to scale. They also require a copy of the device's firmware to perform the attestation, which may not be available due to the IP rights of manufacturers. [16] proposes a more efficient method to attest firmware by taking partial checksums of the flash memory, however, at the cost of the device's availability since the attestation routine must run uninterrupted on the embedded device. Furthermore, flash memory does not provide information about D2D relationships, which is a crucial task in swarms. [15] presents a more suitable SRAM-based approach to detect malicious firmware. The authors propose a classification-based approach to distinguish between normal and malicious firmware for a limited number of anomalous classes and achieve a 96% accuracy in their proposed task. However, this is not practical in a real scenario because an adversary can make any number of changes in the firmware, and the method is not scalable. Furthermore, the entire SRAM is used for this purpose, and we show that we only need a part of it to perform the attestation.

Hardware-based attestation techniques solve the latency problem by relying on TPMs for complex cryptographic computation, and some notable studies include [7–9]. While they have high detection rates and low latency, an assumption of the availability of TPMs on IoT devices will not apply in most cases since most lightweight IoT devices do not have an in-built TPM. Furthermore, replacing the monumental number of devices already deployed in present-day use cases worldwide will be expensive.

Various hybrid attestation techniques have been proposed to find a balance between the complexity of software-based techniques and the hardware requirements of hardware-based techniques [10, 11, 17–19]. These solutions require the attestation routines to run uninterrupted on the IoT device, which hampers the device's regular function and is also not applicable to roving malware. Furthermore, [17] is also susceptible to a single point of failure.

Lastly, some existing works on swarm attestation include FeSA [12], PADS [13], and ESDRA [14]. The authors of [12] propose a distributed attestation protocol that uses federated learning to attest swarms by assigning security and privacy levels to evaluate attestation periods that avoid redundancy during attestation. The proposed method achieves an average

accuracy of 87.7% while preserving the data's privacy and has a lower run-time than software attestation techniques. [13] presents an attestation protocol using the concept of self-attestation that achieves a 2-second evaluation latency for large swarms. While the approach is scalable and has low latency, it limits the availability of the devices during the attestation. Authors of [14] propose a many-to-one attestation scheme based on reputation. However, the proposed method requires computationally intensive tasks on the cluster head.

Looking at the above discussion, it is evident that the existing literature has limitations regarding latency, device availability, scalability, hardware requirements, and the need for firmware copies. Inspired by the work of [15], this paper explores volatile memory (SRAM) as a solution for efficient swarm attestation since it stores information on the firmware's runtime. An SRAM-based approach does not require complicated hardware, a copy of the firmware, and ensures availability during the attestation. Recent studies on SRAM-based fingerprinting [20] showed that the SRAM may be divided into the *data section* and the *stack*. The latter is helpful for hardware fingerprinting since the pseudo-random initializations of the stack are unique to each device. However, the data section, which contains runtime information, has different sizes based on the firmware, has similar behavior for the same firmware loaded on different devices. Therefore, we only use the data section in this paper. Furthermore, a preliminary experiment on D2D relationships shown in Figure 1 highlights that the data collected and sent from an arbitrary node in a swarm (Figure 1 (a)) is embedded in the SRAM contents of the receiving node (Figure 1 (b)). We show that this property helps detect downstream anomalies.

In this paper, we use an MoE architecture [21] for DAE networks to attest swarms.

III. Network and Threat Model

This section introduces the network and threat models considered in the study.

A. Network Model

A swarm attestation routine includes two parties: a verifier and a swarm of provers. The verifier is a trusted entity that sends attestation requests to the provers and has sufficient computational power to analyze the attestation responses. Conversely, provers are lower-power IoT nodes that receive attestation requests from the verifier and respond with their SRAM dump. In the network presented in Figure 2, the verifier sends an attestation request to the swarm's local master node, N_0, which responds with its own SRAM trace and prompts its slave nodes N_1, N_2 and N_3 to respond with their corresponding SRAM dumps directly to the verifier. Therefore, the "swarm response" is a collection of the memory dumps (R_i) of each node N_i in the network. Such a scenario was created to observe different D2D behaviors and may be found in in-vehicle networks (CAN). Also, please

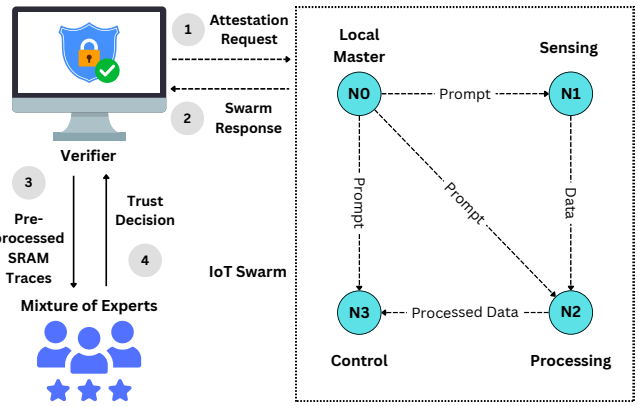


Fig. 2: Network Model.

note that the swarm forms a directed graph where N_0 sends attestation prompts to N_1, N_2, and N_3, while $N_1 - N_3$ share information with each other in the said order. This simple swarm aims to observe the effect of firmware and network anomalies on the SRAM of the anomalous nodes and downstream nodes they communicate with.

IoT device may be categorized into three overall types based on their function. Sense-type nodes collect (or generate) data from sensors, process-type nodes process the data to make decisions, and control-type nodes control output devices. While devices may have more than one function in a practical scenario, we consider the simple (yet complex) IoT swarm, shown in Figure 2, to observe the behavior of each type in a swarm setting. N_1, N_2, and N_3 are sense, process, and control-type nodes, respectively.

In a real-world scenario, a development network may be used to collect data and train attestation schemes. At the same time, a physical twin of the development network (also called a production network) may be deployed on the users' end. Manufacturers may send firmware updates to the production networks. In such cases, any attestation scheme designed on the development network should be repeatable on the physical twin.

B. Threat Model

An adversary may update a node in the swarm with malicious firmware or launch a network attack. There may also be data faults at sensor nodes. Such anomalies may lead to abnormal behavior in downstream nodes.

1. **Firmware anomalies:** An adversary may upload malicious firmware to one or more nodes in the swarm. The anomalous firmware may have a minimal or significant difference in functionality from the authentic firmware. Consequently, depending on the tampered functionality, the adversarial node may affect downstream nodes.
2. **Network attacks:** An adversary may act as a man-in-the-middle and tamper with the data shared between nodes or the SRAM traces sent from the nodes to the verifier. An adversary may launch a denial of

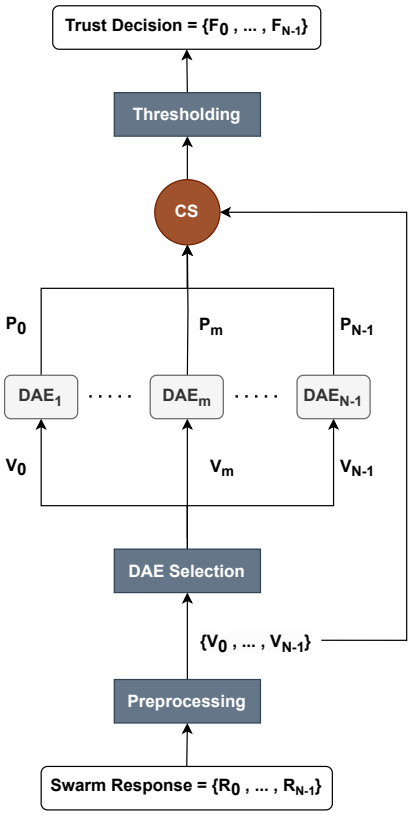


Fig. 3: The proposed MoDAE framework.

service attack, drop messages between nodes, or drop attestation responses from the nodes.

3. **Data faults:** There may be a malfunction at the sense-type nodes, leading to the propagation of faulty data, which has downstream effects. While such an anomaly may not be caused by an adversary, we show that it can be detected during the attestation.
4. **Propagated anomalies:** Firmware anomalies at adversarial nodes, network anomalies, and data faults may create downstream anomalies propagating in the direction of communication. For example, in the swarm shown in Figure 2, faulty data sent from N_1 or dropped messages at N_2 may lead to faulty control signals at N_3.

IV. Proposed Method: SAFE-IoT

We use the MoDAE framework shown in Figure 3 to perform firmware attestation of the swarm. Each node N_i in the swarm is assigned a lightweight DAE_i that learns to reconstruct the SRAM distribution of normal firmware and then detects anomalous behavior. This is achieved in two stages.

A. Training Phase

During the training phase, a verifier in a development network samples training data by sending attestation requests to the master node. The sampled data is preprocessed, and each DAE_i learns to reconstruct a $trainset_i$ perturbed with Gaussian noise of mean and standard deviation of 0 and 1, respectively. Once trained, the DAEs are tested on the $trainset$, and Cosine Similarity (CS) scores are evaluated between the prediction and the input using the equation:

$$CS(x, y) = \frac{x \cdot y}{|x||y|} \tag{1}$$

Where x and y are vectors. Subsequently, the CS scores of the trainset are sorted in ascending order, and the detection thresholds T_i are calculated using the equation:

$$T_i = 0.99 * CS(N_i)_{i,1\%} \tag{2}$$

Where $CS(Ni)_{i,1\%}$ is the CS score of the first percentile datapoint in the sorted CS score array. Such a selection of detection thresholds is intuitive for percentage-like metrics such as CS and provides a consistent way of selecting accurate thresholds. Please note that while Mean Squared Error (MSE) is typically used as the thresholding metric in anomaly detection tasks, CS was chosen in this study instead due to the simplicity of selecting thresholds (regardless of the node) using Equation 2. In addition, being a similarity metric, T_i is a lower bound of normal behavior. The attestation parameters, i.e., (DAE, T), are then deployed onto the verifier of a production network.

B. Attestation Phase

The procedure for attestation is shown in Algorithm 1. The verifier loads the DAE_i and T_i for all N_i in the swarm and initializes the swarm response V and trust decision F. An attestation $request$ is then sent to the local master N_0, and each N_i in the swarm responds to the verifier with its SRAM dump, which is preprocessed and updated into V. The nodes' responses are assigned to their respective DAE, which subsequently reconstruct the input traces. Similarity scores are evaluated between the predictions and the preprocessed traces using Equation 1 for each node, which are then compared to the previously assessed thresholds, T. If N_i's score lies above T_i, it is flagged as *"S"* (safe); otherwise, it is flagged as *"A"* (anomalous). In the case where a node does not respond with its memory dump within the timeout from receiving the attestation request, it is flagged *"NR"* (no response). In this way, the trust decision F is obtained for the entire swarm.

V. Experimental Setup

This section details the experimental testbed and the collected dataset.

A. Hardware, Communication, and Software

1) **Hardware:** All nodes in the swarm shown in Figure 2 are variants of Arduino and Elegoo UNO rev3, each with an ATmega328P microcontroller and 2KB SRAM. The verifier device is a personal computer with a 14th-generation Intel i9 processor, 64 gigabytes of DRAM, and an Nvidia RTX 4080 GPU.

Algorithm 1: Attestation algorithm

```
// Load dependencies
DAE, T ← load(swarm)
V ← [[], ..., []] // numNodes empty arrays
F ← [0, ..., 0]
score ← [0, ..., 0]
// Request - Response
request(N_0)
while time ≤ timeout do
    V[i] ← preprocess(receive(N_i))
// Evaluate the swarm response
i ← 0
while i ≤ numNodes do
    if V[i] == [] then
        F[i] ← NR // No response
    else
        pred ← DAE[i].predict(V[i])
        score[i] ← CS(pred, V[i])
        if score[i] > T[i] then
            F[i] ← S // Safe
        else
            F[i] ← A // Anomaly
    i ← i + 1
// Return trust decision
return([F_0, ...., F_{numNodes−1}])
```

2) **Communication Protocols:** N_1, N_2, and N_3 communicate with each other using the Inter-integrated Circuit (I2C) protocol, while N_0 sends attestation prompts to them using the Serial Peripheral Interface (SPI) protocol. Each node sends its SRAM trace to the verifier via the serial monitor. Since N_0 broadcasts a prompt to all nodes, the nodes can simultaneously print their memories onto the serial monitor, which takes about 1.2 seconds for the whole swarm on average and is the same for an even larger number of connected devices. The read operation for the verifier takes minimal time, making the attestation process very efficient. Please note that the protocols used in the network are purely for ease of data collection using the selected devices and have no effect on the attestation and detection rates of *SAFE-IoT*. In a real application, devices may communicate with each other via wireless communication channels such as Bluetooth, Bluetooth Low Energy (BLE), WiFi, Zigbee, etc. *SAFE-IoT* can be applied to SRAM data collected from any wired or wireless network.
3) **Software:** The firmware uploaded onto the nodes is designed using C++ in Arduino IDE 2.3.2, while the codes deployed on the verifier for data collection, statistical analysis, and the attestation framework are developed in Python 3.8. The main Python libraries used in this study are pyserial 3.5, numpy 1.26.4, pandas 2.2.2, tensorflow 2.16.1, and keras 3.3.3.

B. Firmware

The Input (I/P), main Functionality (F), Output (O/P), Attestation Sub-routine (ASR), and data thresholds (d) of normal and anomalous firmware are shown in Table I.

N_0 is the swarm's master device. It receives attestation requests from the verifier and prompts the slave nodes to execute their respective ASR. The anomalous variant of this node emulates a firmware anomaly at N_0, given its added functionality of generating three random integers. This anomaly poses a relatively tricky challenge during attestation, but it does not affect the behavior of the slave nodes.

N_1 is a sense-type node that generates six floating-point numbers in unique ranges to emulate variant sensor data. In its anomalous variant, N_1 generates data in an extended range. This emulates a firmware anomaly at N_1 and a network attack or possible data fault when observed from N_2 and N_3; the latter is challenging since the variables may or may not lie simultaneously in their original ranges. However, it does not affect N_0 since there is no communication in that direction.

N_2 is a process-type node that receives data from N_1 and generates a six-byte binary control signal using which N_3 controls its LEDs. In its anomalous variant, N_2 drops the data it receives from N_1 and generates a random control signal that may contain abnormal values. The anomalous variant thus emulates a firmware anomaly at N_2 and a possible network attack or data fault when observed from N_3. However, it does not affect N_0 and N_1 since there is no communication in those directions.

N_3 is a control-type node that controls six output LEDs based on the control signal received from N_2. In the anomalous variant, N_3 drops the actual control signal and randomly turns the LEDs on or off, thus emulating a firmware anomaly. It does not affect any other node in the swarm.

C. Dataset

The dataset was collected from the experimental setup discussed in prior subsections. It covers thirteen scenarios, as shown in Table II, of which, D_1, D_2, P_1 and P_2 are safe network scenarios while AN_0 - AN_{0123} are anomalous ones. Data for D_1 and D_2 is collected from two independent initializations of the development network, while that for P_1 and P_2 is collected from two physical twins of the development network. The anomalous scenarios are labeled AN_i, representing a primary firmware anomaly at N_i, and there may be more than one primary anomaly as seen in AN_{12}, AN_{13}, AN_{23}, AN_{123} and AN_{0123}. The primary and secondary anomalies are also highlighted in the table. Each scenario has four hundred samples of SRAM dumps at each node. The collective memory dump at any given index (or instance in time) represents a synchronized network state since the SRAM data is collected after the nodes complete their message exchanges.

TABLE I: Comparison of input data, output data, functionality, ASR, and .data section lengths between normal and malicious firmware samples.

Node		Normal variant	Anomalous variant
N_0 (Master)	I/P	Receives attestation requests from the verifier.	Receives attestation requests from the verifier.
	F	Prompts slave devices to print SRAM contents.	Prompts slave devices to print SRAM contents. Generates three random integers.
	O/P	Sends a one-byte prompt to N_0, N_1, N_2.	Sends a one-byte prompt to N_0, N_1, N_2.
	ASR	Prints the SRAM contents onto the serial monitor.	Prints the SRAM contents onto the serial monitor.
	d	**191**	**195**
N_1 (Sense)	I/P	Receives a one-byte prompt from N_0.	Receives a one-byte prompt from N_0.
	F	Generates six floating point numbers in fixed, unique ranges.	Generates six floating point numbers in a larger range.
	O/P	Sends 32 bytes to N_2	32 bytes (containing the generated data) to N_2
	ASR	Prints the SRAM contents onto the serial monitor.	Prints the SRAM contents onto the serial monitor.
	d	**450**	**438**
N_2 (Process)	I/P	Receives a one-byte prompt from N_0. Receives thirty two bytes from N_1 .	Receives a one-byte prompt from N_0. Receives thirty two bytes from N_1.
	F	Extracts six floating point numbers from the received bytes. Generates a six-byte control signal	Discards received data. Generates a random six-byte control signal.
	O/P	Sends a six-byte control signal to N_3.	Sends a six-byte control signal to N_3.
	ASR	Prints the SRAM contents onto the serial monitor.	Prints the SRAM contents onto the serial monitor.
	d	**516**	**414**
N_3 (Control)	I/P	Receives a one-byte prompt from N_0. Receives six bytes from N_2.	Receives a one-byte prompt from N_0. Receives six bytes from N_2.
	F	Extracts the control signal from the received bytes. Controls six output LEDs using the extracted signal.	Discards the received control signal. Controls six output LEDs at random.
	O/P	N/A	N/A
	ASR	Prints the SRAM contents onto the serial monitor.	Prints the SRAM contents onto the serial monitor.
	d	**406**	**386**

TABLE II: Network scenarios in the dataset.

S.No.	Scenario	Type	Samples/ node	Primary Anomaly	Secondary Anomaly
1	D_1	S / train	400	-	-
2	D_2	S / train	400	-	-
3	P_1	S / test	400	-	-
4	P_2	S / test	400	-	-
5	AN_0	A / test	400	N_0	-
6	AN_1	A / test	400	N_1	N_2, N_3
7	AN_2	A / test	400	N_2	N_3
8	AN_3	A / test	400	N_3	-
9	AN_{12}	A / test	400	N_1, N_2	N_3
10	AN_{13}	A / test	400	N_1, N_3	N_2
11	AN_{23}	A / test	400	N_2, N_3	-
12	AN_{123}	A / test	400	N_1, N_2, N_3	N_2
13	AN_{0123}	A / test	400	All	-

D. Preprocessing

Each SRAM sample consists of 2048 integer equivalents (i.e in the range [0,255]) of the hexadecimal bytes stored in the 2KB SRAM of each device. For further analysis, the data of each node is truncated to the data thresholds of that node's authentic firmware regardless of the network's anomalous state. This is because normal data and the data threshold of the authentic firmware are the only prior information we have while developing the attestation framework. The data is then scaled down by a factor of 255 to bring it to a [0,1] range.

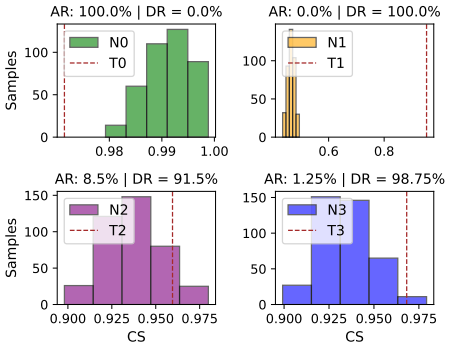
E. Denoising Autoencoder Experts

Two types of DAEs are used. DAE_0 and DAE_3 have a convolutional encoder with 16 filters and a fully connected decoder of 50 neurons, each followed by a dropout of 0.1, while DAE_1 and DAE_2 use a fully connected encoder and decoder of 96 neurons, each followed by a dropout of 0.2. All layers are activated using the Rectified Linear Unit (ReLU) activation function. The DAEs are optimized to minimize MSE using the Adam optimizer at a 0.01 learning rate for 100 epochs. Each model occupies nearly 1MB of verifier memory.

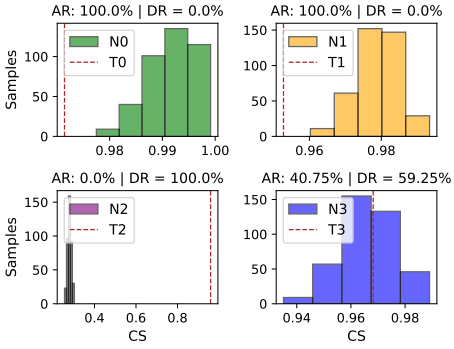
F. Metrics

Since *SAFE-IoT* is posed as a binary classification task (safe vs. anomalous behavior), we use the true negative rate as the attestation rate (AR) of the anomaly detection task and the true positive rate as the anomaly detection rate (DR). The metrics may be evaluated using the equations:

$$AR = TN/(TN + FP) \tag{3}$$



(a) AN_1



(b) AN_2

Fig. 4: Performance of a purely similarity-based approach in difficult scenarios.

$$DR = TP/(TP + FN) \tag{4}$$

Where TN, TP, FN, and FP are true negatives, true positives, false negatives, and false positives, respectively.

G. Preliminary Analysis

It is valuable to show some preliminary statistical results to help understand the difficulty of different detection tasks. Upon receiving an attestation request, each N_i responds with its SRAM dump R, which the verifier then truncates to obtain the data section as:

$$v_i^S = (b_0, b_1, ..., b_d)_i^S \tag{5}$$

Where b_j is the j^{th} SRAM byte, d is the data threshold of N_i, and S is the scenario. We compute the golden reference G_i of N_i from the concatenated distribution $D_1||D_2$ using the equation:

$$G_i = (b_{0,avg}, b_{1,avg}, ..., b_{d,avg})_i^{D_1||D_2} \tag{6}$$

We now perform a sample-wise comparison of data from each scenario with the node-wise G_i by selecting a suitable threshold for separating normal and anomalous behavior. Attestation is achieved using thresholds T_i selected for each N_i using Equation 2; however, in this case, the CS is evaluated between each sample and G_i instead of the input and output of a DAE. The AR and DR for two difficult tasks AN_1 and AN_2 are shown in Figure 4. As Figure 4 (a) shows, the DR of anomalies at N_2 is 91.5% while that at N_3 is 98.75%. The performance is much worse in the case of AN_2, as shown in Figure 4 (b); the downstream anomalous samples at N_3 are detected at only 59.25%. The poor performance in D2D propagated anomalies can be explained by the loss of useful information about distributions after averaging that machine learning approaches may otherwise learn. In addition to the above metrics, we also attempted to use other distance metrics, such as Jensen-Shannon Divergence and Kullback-Leiber Divergence, which were not fruitful to the study. However, the preliminary experiments show differences in the SRAM distributions for various node-level and downstream cases.

TABLE III: Summary of AR/DR in all test scenarios.

S.No.	Scenario	N0	N1	N2	N3
1	P_1	100	100	100	100
2	P_2	100	98.25	99.5	100
3	AN_0	100	98.25	100	100
4	AN_1	100	100	98	100
5	AN_2	100	99.5	100	96
6	AN_3	100	99	100	100
7	AN_{12}	100	100	100	96.25
8	AN_{13}	100	100	98.25	100
9	AN_{23}	100	99.25	100	100
10	AN_{123}	100	100	100	100
11	AN_{0123}	100	100	100	100

TABLE IV: Comparison of latency (in seconds for 1000 devices) and average detection rate with related works.

S.No.	Category	*SAFE-IoT*	[15]	[12]	[13]	[14]	[4]	[10]
1	ASR	1.2	1.6	-	-	-	-	-
2	Evaluation	10^{-1}	4	0.6	2	0.4	10^3	10^2
3	AR/DR	99.5%	96%	87.7%	-	-	-	-

VI. Experimental Results

This section presents the results of *SAFE-IoT* on firmware attestation, robustness, and latency.

A. Detection rates on the SAFE-IoT dataset

The AR and anomaly DR scores of our best-performing initialization of *SAFE-IoT* on the dataset are shown in Figure 5 and summarized in Table III. The MoDAE has a 100% DR on primary firmware anomalies in all scenarios. Furthermore, it has significantly higher performance than the results shown in Figure 4 on downstream anomalies, with 98.25% and 100% DR at N_2 and N_3 respectively in AN_1 and 96.25% DR at N_3 in AN_2. Furthermore, the performance is consistent in scenarios with multiple firmware anomalies (AN_{12}-AN_{0123}). There are instances of false positive anomalies in P_1 and P_2, which occur since the detection thresholds T are selected very close to the lower bound of CS scores on the training data (as seen from Equation 2) to detect D2D anomalies, data faults, and perturbation-type anomalies, However, in use cases where there is flexibility in terms of attestation rate,

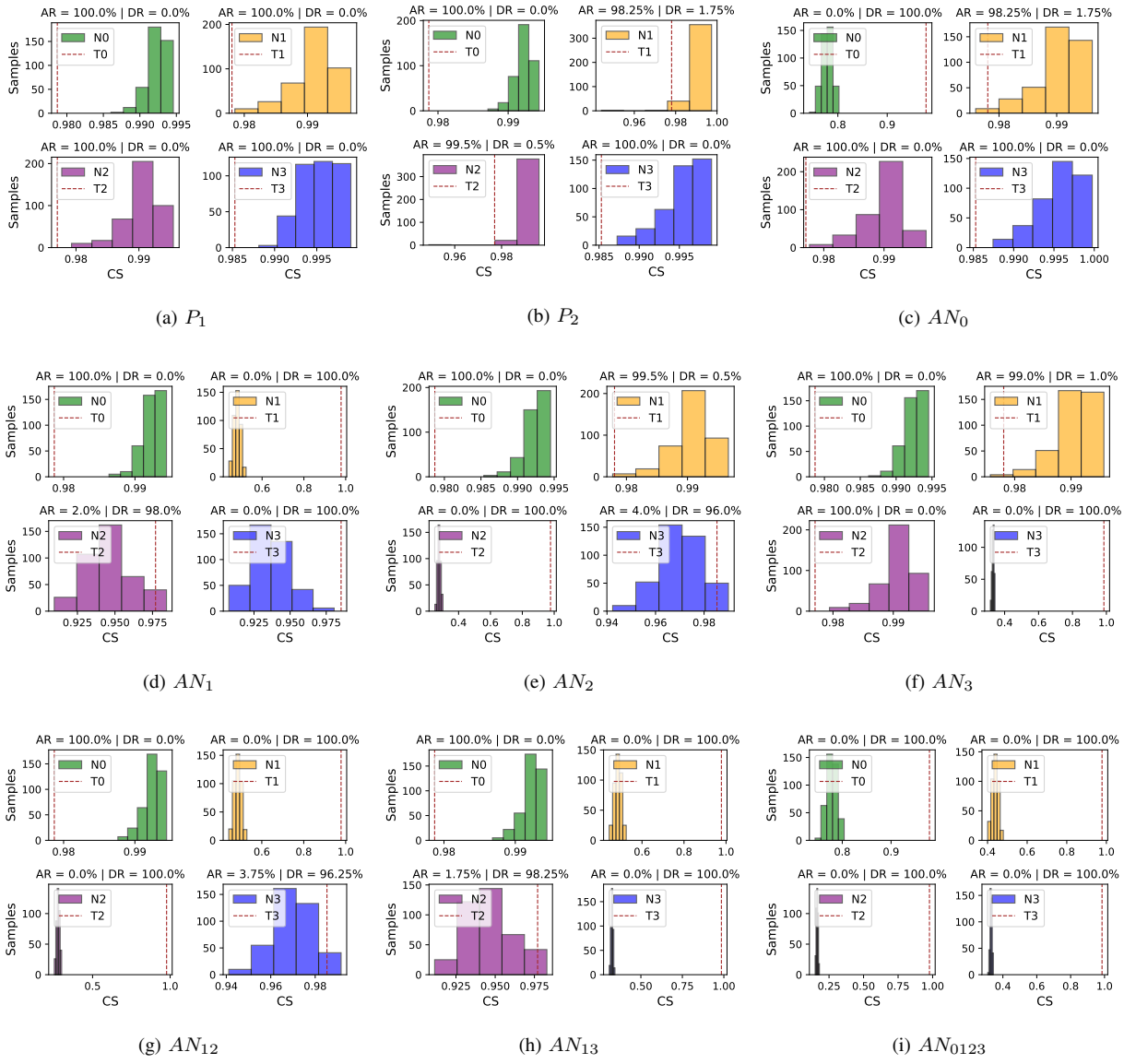


Fig. 5: Performance of *SAFE-IoT* on various test cases.

a more lenient threshold may be selected by using a 0.98 scaling factor in Equation 2. Overall, the MoDAE has a 98+% AR on safe behaviors, 100% DR on anomalous firmware and a 95+% DR on downstream anomalies. On average, the proposed approach has a 99.5% AR/DR, which is an improvement over past works, as shown in Table IV.

B. Robustness against tampering

To show robustness against data faults and tampering, the MoDAE was tested on a patched dataset comprising normal data perturbed with noisy patches of varying sizes. The results of this experiment are shown in Figure 6. The MoDAE has a 95% DR on average at around 8 bytes (two floating point variables) of random noise across all nodes. Adjusting the threshold to be closer to the lower bound of CS scores on the train data helps improve the performance in this task; however, it is at the expense of the AR on authentic firmware. Thus, the present setting best balances consistent performance on normal and anomalous behavior.

C. Latency analysis

The latency of the ASR and evaluation are shown in Table IV. The entire swarm takes, on average, 1.2 seconds to complete one request-response routine for 2kB of SRAM contents, which corresponds to the communication overhead. Since the ASR runs on all nodes simultaneously, it has nearly the same latency for a larger number of devices. The MoDAE has an evaluation latency (preprocessing + reconstruction + threshold) of the order 10^{-4} seconds per device. Simulating a scenario with 1000 such devices scales the latency to the

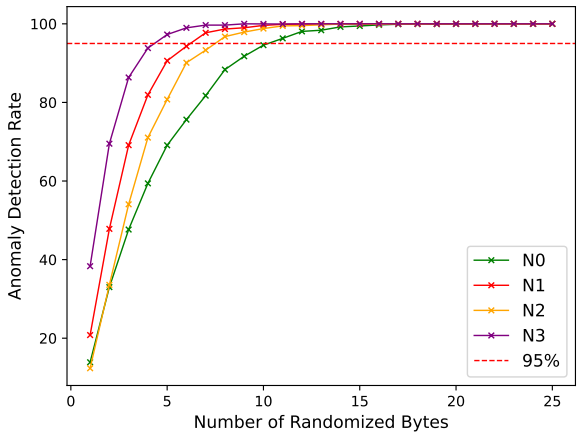


Fig. 6: Performance on byte-incremental randomization of normal behavior.

order 10^{-1} seconds, faster than past works at the same swarm size.

D. Limitations

While *SAFE-IoT*'s MoDAE framework has high detection capabilities, the DR varies significantly in the case of downstream effects depending on parameter initialization. This makes it difficult to reliably and intuitively select an accurate model. Taking AN_1 as an example, the DR at N_2 can be as low as 90% in some initialization despite having nearly 100% DR on malicious firmware. Future works may explore using graph learning as an alternative due to its robustness in handling graph-structured data.

VII. Conclusion

This article proposed a novel technique called SAFE-IoT for firmware attestation in IoT swarms using volatile memory and denoising autoencoders organized as an MoE. It also presented the first dataset for SRAM-based attestation of IoT swarms. A preliminary statistical analysis was conducted to highlight the difficulty of different anomalous behaviors. The proposed method could attest authentic firmware at 99+% and detect 100% firmware anomalies and 95+% propagated anomalies, respectively. Robustness experiments showed 95% detectability at 8 bytes against added noise. The proposed method took 1.2 seconds for the request-response routine and 10^{-4} seconds per device to evaluate the entire swarm.

References

[1] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[2] M. Khodari, A. Rawat, M. Asplund, and A. Gurtov, "Decentralized firmware attestation for in-vehicle networks," in *Proceedings of the 5th on Cyber-Physical System Security Workshop*, 2019, pp. 47–56.

[3] L. Ilascu, "When their firmware is vulnerable, its up to you to protect your smart devices," *Accessed: May*, vol. 5, 2019.

[4] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla, "Swatt: Software-based attestation for embedded devices," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004.* IEEE, 2004, pp. 272–282.

[5] A. Seshadri, M. Luk, A. Perrig, L. Van Doorn, and P. Khosla, "Scuba: Secure code update by attestation in sensor networks," in *Proceedings of the 5th ACM workshop on Wireless security*, 2006, pp. 85–94.

[6] A. Seshadri, M. Luk, and A. Perrig, "Sake: Software attestation for key establishment in sensor networks," in *Distributed Computing in Sensor Systems: 4th IEEE International Conference, DCOSS 2008 Santorini Island, Greece, June 11-14, 2008 Proceedings 4*. Springer, 2008, pp. 372–385.

[7] S. Agrawal, M. L. Das, A. Mathuria, and S. Srivastava, "Program integrity verification for detecting node capture attack in wireless sensor network," in *Information Systems Security: 11th International Conference, ICISS 2015, Kolkata, India, December 16-20, 2015. Proceedings 11*. Springer, 2015, pp. 419–440.

[8] H. Tan, W. Hu, and S. Jha, "A tpm-enabled remote attestation protocol (trap) in wireless sensor networks," in *Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, 2011, pp. 9–16.

[9] W. Yan, A. Fu, Y. Mu, X. Zhe, S. Yu, and B. Kuang, "Eapa: Efficient attestation resilient to physical attacks for iot devices," in *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, 2019, pp. 2–7.

[10] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "Tytan: Tiny trust anchor for tiny devices," in *Proceedings of the 52nd annual design automation conference*, 2015, pp. 1–6.

[11] M. N. Aman and B. Sikdar, "Att-auth: A hybrid protocol for industrial iot attestation with authentication," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5119–5131, 2018.

[12] B. Kuang, A. Fu, Y. Gao, Y. Zhang, J. Zhou, and R. H. Deng, "Fesa: Automatic federated swarm attestation on dynamic large-scale iot devices," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[13] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, and S. Ranise, "Pads: Practical attestation for highly dynamic swarm topologies," in *2018 International Workshop on Secure Internet of Things (SIoT)*. IEEE, 2018, pp. 18–27.

[14] B. Kuang, A. Fu, S. Yu, G. Yang, M. Su, and Y. Zhang, "Esdra: An efficient and secure distributed remote attestation scheme for iot swarms," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8372–8383, 2019.

[15] M. N. Aman, H. Basheer, J. W. Wong, J. Xu, H. W. Lim, and B. Sikdar, "Machine-learning-based attestation for the internet of things using memory traces," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20 431–20 443, 2022.

[16] B. Chen, X. Dong, G. Bai, S. Jauhar, and Y. Cheng, "Secure and efficient software-based attestation for industrial control devices with arm processors," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 425–436.

[17] K. Eldefrawy, G. Tsudik, A. Francillon, and D. Perito, "Smart: secure and minimal architecture for (establishing dynamic) root of trust." in *Ndss*, vol. 12, 2012, pp. 1–15.

[18] M. N. Aman, M. H. Basheer, S. Dash, J. W. Wong, J. Xu, H. W. Lim, and B. Sikdar, "Hatt: Hybrid remote attestation for the internet of things with high availability," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7220–7233, 2020.

[19] M. N. Aman, M. H. Basheer, S. Dash, A. Sancheti, J. W. Wong, J. Xu, H. W. Lim, and B. Sikdar, "Prom: passive remote attestation against roving malware in multicore iot devices," *IEEE Systems Journal*, vol. 16, no. 1, pp. 789–800, 2021.

[20] V. Kohli, M. N. Aman, and B. Sikdar, "An intelligent fingerprinting technique for low-power embedded iot devices," *IEEE Transactions on Artificial Intelligence*, 2024.

[21] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE transactions on neural networks and learning systems*, vol. 23, no. 8, pp. 1177–1193, 2012.