SRAM and Generative Network-based Physical Fingerprinting for Trust Management in the Internet of Things

Varun Kohli†, *Student Member, IEEE*, Muhammad Naveed Aman‡, *Senior Member, IEEE*, Biplab Sikdar†, *Senior Member, IEEE*
†Department of Electrical and Computer Engineering, National University of Singapore, Singapore.
‡School of Computing, University of Nebraska-Lincoln, USA.

***Abstract*—Recent advances in the Internet of Things (IoT), machine learning, and edge computing have led to the development of paradigms such as smart cities, smart grids, smart healthcare, and intelligent transportation systems as efficient and cost-effective solutions. Subsequently, there has also been an increase in the number of connected devices, ranging from high-power computers to low-power microcontrollers and sensors. The multi-layered and complex structure of the IoT creates a vast surface of vulnerabilities. Cyber threats such as proxy attacks are prevalent in ubiquitous resource-constrained IoT devices giving rise to the need for practical device fingerprinting algorithms. Existing works are based on network activity deep learning-based classification methods. However, attackers can mimic network activity, and classification models must be retrained on the data of new anomalies. This paper solves these issues by proposing a lightweight, and intelligent physical fingerprinting algorithm and the corresponding mutual authentication protocol using initial power-up Static Random Access Memory (SRAM) states and generative networks. A generative network is trained to reconstruct SRAM fingerprints of an authorized device during the registration phase and an anomaly threshold is selected. The proposed technique reliably fingerprints registered devices, and can detect proxy devices of identical architectures from the same and different manufacturers with high accuracy. Since the algorithm does not require attacker data during the training phase, it is a self-sufficient and low-cost solution. The method has a latency of 2.114 seconds per device and security analysis of the proposed protocol proves its security against proxy and de-synchronization attacks.**

***Index Terms*—Internet of Things, Fingerprinting, Anomaly Detection, Deep Learning, Trust Management**

I. Introduction

The Internet of Things (IoT) is an aggregation of low to high-power devices that collect, process, or share data with each other via a communication network such as the Internet. Various IoT paradigms including smart cities, smart grids, smart healthcare, smart agriculture, smart industry, smart supply chain, and intelligent transportation systems have been enabled over the past decade by taking advantage of sensing [1], processing technologies such as cloud and edge computing [2], machine learning [3], and 5G/6G telecommunication [4] to make processes cheaper, reliable and efficient [5]. However, due to their distributed, and multi-layered architecture, IoT networks are prone to cyber attacks targetting network availability, user privacy, and device or data integrity [6].

Threats to device integrity such as proxy attacks are prevalent in low-power, low-cost devices such as sensors and microcontrollers, and timely detection of compromised nodes is a difficult and important challenge. Although a few efficient network-based fingerprinting algorithms have been developed [7, 8], such methods are vulnerable to attacks that mimic the device's network activity to avoid detection. In such cases, it is necessary to evaluate the devices at a much lower level to achieve trust in the network. Distributed remote attestation protocols using Physically Unclonable Functions (PUF) and blockchain show promise in this regard [9]. A few other studies propose PUF-based protocols for malware mitigation [10] and mutual authentication [11]. The authors in [12] showed that the uniqueness of initial SRAM states in RFID tags due to manufacturing variability can be used to generate fingerprints. Similarly, another study on radio frequency transmitters identifies a similar property for wireless transmitters and proposes a deep learning-based classifier for a fixed number of devices [13]. However, problems may arise when a new device is introduced into the network. In this case, the singular detection network must be retrained to accommodate the new device. Furthermore, classification schemes are limited in their ability to detect unknown anomalies owing to the necessity for prior knowledge of the attack in the form of data. Studies on intelligent low-cost anomaly detection in vehicular networks solve this problem with a prior anomaly-data-less classification scheme using generative networks and reconstruction error thresholding to perform a "one vs any" classification [14, 15].

It is apparent from the above discussion that the existing techniques are vulnerable to cyber attacks that mimic network activity, and threaten physical integrity. There are also problems when previously unknown devices enter the network. To solve these issues, this paper proposes an intelligent fingerprinting algorithm and the corresponding mutual authentication protocol to manage the trustworthiness of resource-constrained IoT devices. The fingerprinting algorithm is built

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.

TABLE I: Notations used in this paper.

Notation	Description
ID_i	Identity of IoT device i
GN_i	Generative network of IoT device i
T_i	Anomaly threshold of IoT device i
$Param_i$	Algorithm parameters GN_i and T_i
RE	Reconstruction error
MSE	Mean squared error
N	Nonce
k_{AV}	Secret symmetric key between device A and V
M_k	Message M is encrypted using key k
H(X)	Hash of X
$\|\|$	Concatenation operator
P	Probability of brute force attack

on two underlying concepts: the unique pseudo-randomness of the SRAM and the anomaly detection capability of generative networks. Furthermore, it is a self-sufficient method since it does not require anomalous device data prior to the testing phase. The contributions of this paper are as below.

1) An intelligent and secure algorithm to verify resource-constrained IoT devices using their initial power-up SRAM states as a physical fingerprint. Given an SRAM size, the proposed algorithm can decipher between devices of identical architectures manufactured by the same as well as different manufacturers. The algorithm consists of simplistic two-layer generative networks and simple mathematical operations.
2) A mutual authentication-based protocol is proposed.
3) Security analysis is conducted to verify the security of the proposed algorithm and protocol.
4) Experimental evaluation of the proposed fingerprinting technique on actual hardware.

Table I provides a short description of the notations used in this paper. The remainder of the paper is organized as follows: Section II discusses the bases of the proposed fingerprinting algorithm. Section III describes the system model considered in this paper. The proposed algorithm and protocol are presented in Sections IV and V, respectively. Section VI details the experimental setup used following which the results are presented in Section VII. Security analysis and the proof are done in Section VIII and the paper is concluded in Section IX.

II. Bases of the Proposed Technique

The proposed algorithm discussed in Section V is built on two underlying concepts: the unique initial pseudo-random power-up SRAM states and the anomaly detection properties of generative networks. This section discusses these two concepts in detail.

A. Initial Pseudo-random Power-up SRAM States

Static Random Access Memory (SRAM) is a type of volatile memory. Unlike Dynamic Random Access Memory (DRAM), which must be periodically refreshed due to high volatility, an SRAM retains data for as long as it is powered up because of its latched circuit design. Figure 1 shows a typical 6 transistor (6T) SRAM cell which comprises 6 MOSFETs numbered $M1 - M6$. $M1 - M4$ form two cross-coupled inverters that store a single bit value, while $M5$ and $M6$ control access to a storage cell for read and write operations. The World Line (WL) controls the hold/read/write operations while the two Bit Lines (BL and $\overline{BL}$) carry the data. Q and $\overline{Q}$ form states ($\overline{Q}Q$) driven by the inverters, which take on the values 00, 01, or 10 for power off, data bit 0, and data bit 1, respectively. States 01 and 10 are stable inverter states and either of the two is selected based on a threshold mismatch occurring at the time of manufacturing, thermal noise, and shot noise. In the case of heavy threshold mismatch, the SRAM cell is biased towards one of the two states and not affected by noise. When the threshold mismatch is low, noise is sufficient to change the state. These tendencies result in power-up SRAM states which are unique to each device and cannot be feasibly replicated. This is supported by a study conducted on RFID tags that showed the applicability of initial SRAM states of RFID tags as a physical fingerprint [12] and forms the first basis of this paper.

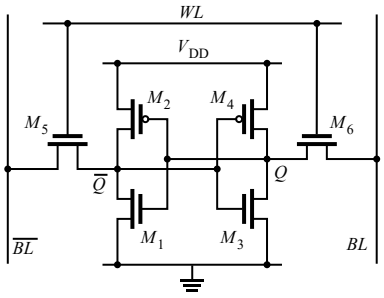


Fig. 1: A typical 6T SRAM cell.

Furthermore, the SRAM is divided into two broad sections: .data and stack, based on the type of variables declared in the firmware. The .data section occupies the initial bytes of the SRAM and stores the global-initialized and uninitialized variables, while the stack stores the run-time data including local and dynamically allocated variables, and function and interrupt return addresses. For a given firmware and device type, the .data section is the same no matter what device it is deployed on. Since the initial power-up SRAM state of a device loaded with a certain code does not contain run-time data, the pseudo-random initialization located in the stack must be used instead. Thus, this paper exploits the SRAM stack for the purpose of fingerprinting.

B. Generative Networks, Reconstruction Error, and Anomaly Thresholds

Generative networks (GN) are a class of unsupervised deep neural networks that learn patterns in the input distribution and generate an output distribution mapped to the input. A widely known example of GNs is the Generator Adversarial Network (GAN) [16], consisting of a generator and discriminator network playing a two-player zero-sum minimax game. These networks have been popular in anomaly detection studies [17].

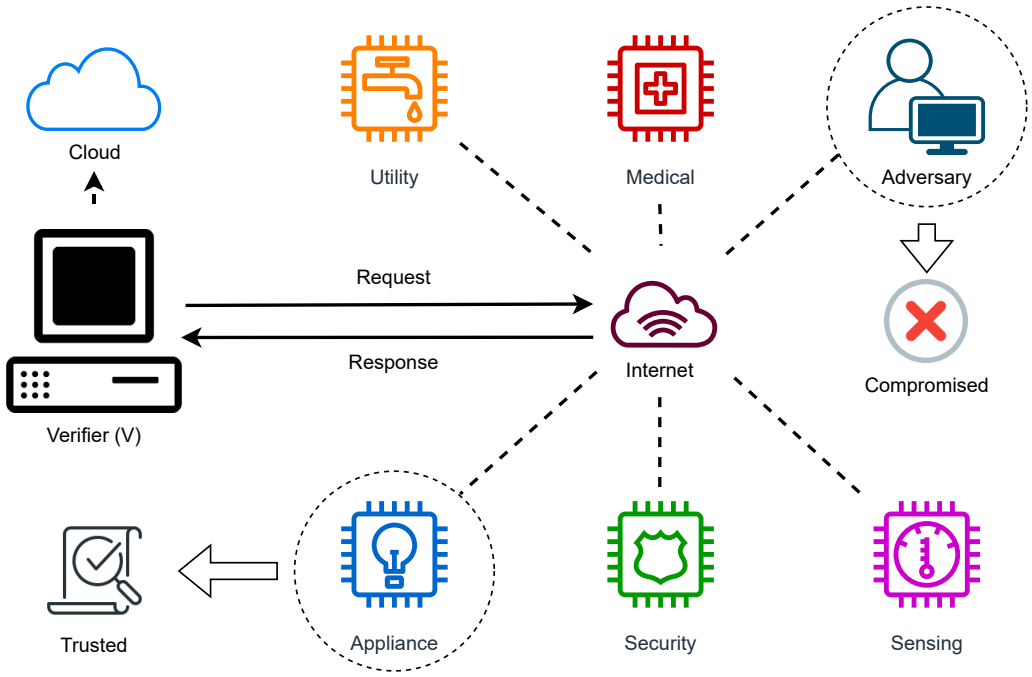


Fig. 2: System model.

However, in this paper, our goal is to use a generator network to develop a lightweight solution that can be trained and deployed on low-capability devices.

In the case of reconstruction, a single GN may recreate the input data in the output. Therefore, the Reconstruction Error (RE), defined as the Mean Squared Error (MSE) between the original and reconstructed trace, measures the closeness of a reconstructed trace to the original input. It also acts as a simplified decision variable that can be analyzed to give a decision on a possibly complicated input distribution. Equations 1 and 2 are used for calculating the RE and MSE, respectively.

$$RE(input) = MSE(input, GN(input)), \tag{1}$$

$$MSE(x_1, x_2) = \frac{1}{n}\sum_{i=1}^{n}(x_{1i} - x_{2i})^2, \tag{2}$$

where $GN(input)$ is the reconstructed input , and x_1 and x_2 are two vectors of size n. Since the quality of reconstruction is inversely proportionate to its RE, a GN is trained to minimize its RE on the training distribution during the training phase. Furthermore, the highest RE for the training distribution corresponds to the maximum permissible error for the input distribution and may be used as a threshold, which we can call the anomaly threshold (T). It is mathematically represented as

$$T = argmax(RE_{input}), \tag{3}$$

where RE_{input} is the set of RE values for all samples of the input distribution derived from its trained GN. The significance of this metric lies in its anomaly detection capability, i.e., when the network faces an input belonging to another distribution, the resulting RE should lie above T. However, when an authentic sample is used, the resulting RE is expected to lie below T. It thus enables us to do a "one vs any" classification statistically, without the need to train on the anomalous data, making this approach self-sufficient. This has been shown in recent studies on intrusion detection in vehicular networks [14, 15] and is a handy observation constituting the second basis of the proposed fingerprinting algorithm.

In the context of device fingerprinting, since the pseudo-random power-up SRAM states of a device are unique to itself, a GN trained on one device's SRAM fingerprint should have lower RE on samples from the same device, and higher RE on samples from another device. RE is thus a simplified decision variable derived from an SRAM trace that can be used to fingerprint a device. As the results of this paper show, an RE below the anomaly threshold is sufficient to fingerprint a device, while one above the threshold is sufficient to show an anomaly.

III. System Model

Figure 2 shows the system model considered in this paper. It is a typical IoT network comprising diferent types of devices connected to a verifier over the internet. We assume that each IoT device has a preshared secret key k with the verifier. Furthermore, they also have a preshared set of emergency nonces that are used in the event of synchronization errors (discussed in Section V). The two overall entities in the network are:

1) Prover: The prover (P) is a resource-constrained IoT device in the network that must be fingerprinted. Upon receiving a fingerprint "request" from the verifier, P sends its SRAM fingerprint as a "response" via a wireless or wired channel depending on the device. A prover may be an authorized registered device, or an unauthorized device (adversary).

2) Verifier: The verifier (V) is a secure and trusted computer that sends fingerprinting requests to the provers via a wireless or wired channel, collects their SRAM fingerprints, and verifies them using the locally deployed fingerprinting algorithm via the proposed protocol. To ensure trust in the network, the verifier periodically initiates one run of the protocol for each registered device. Once identified, it also stores relevant information to mitigate the adversary's effect on the network.

IV. Proposed Fingerprinting Algorithm

The proposed fingerprinting technique comprises of two phases: registration and authentication.

A. Registration phase

An IoT network may consist of a number of IoT devices that must first be registered on the fingerprinting algorithm through the registration phase as depicted in Figure 3. The goal of this phase is to obtain the trainable GN weight parameters (GN_i) and the anomaly threshold (T_i) for a device ID_i. These parameters are collectively called the algorithm parameters ($Param_i$) of the device. To register a device, the verifier first creates a fingerprinting dataset by sampling a number of initial power-up SRAM states. The dataset is preprocessed and used to train the reconstruction network GN_i which is then used to select T_i using Equation 3. The resulting parameters are

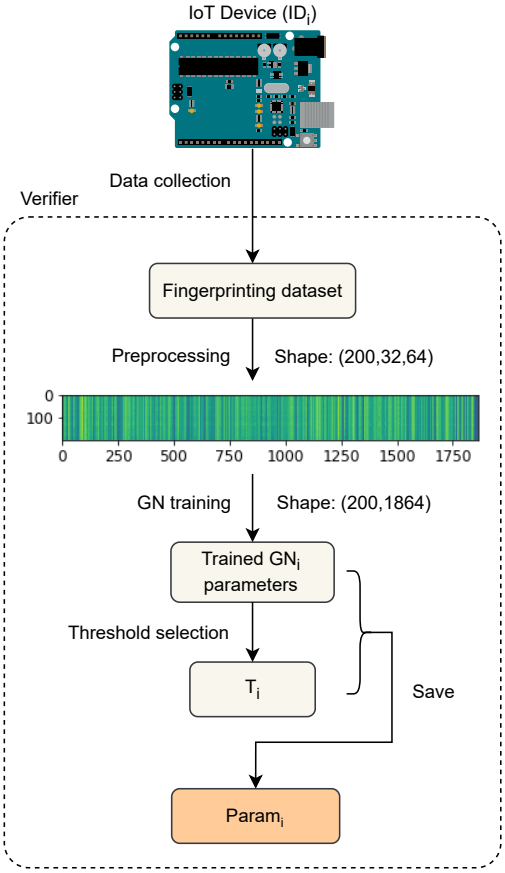


Fig. 3: Registration phase.

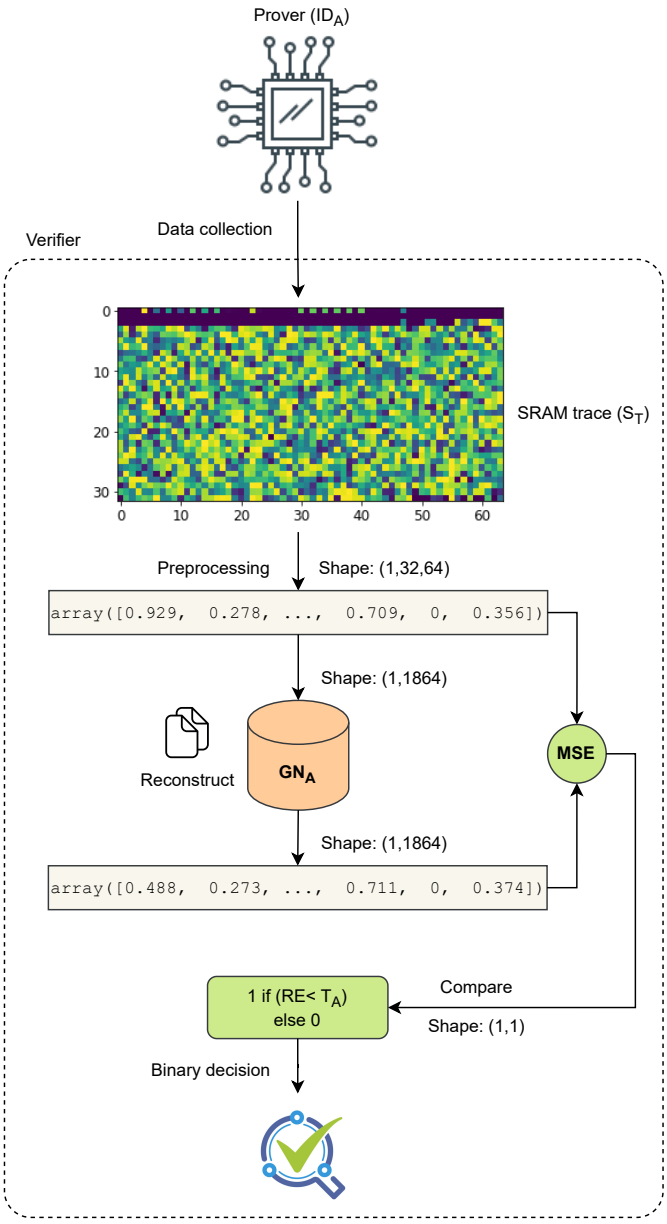


Fig. 4: Authentication phase ($Param_A$).

saved in the verifier's memory as $Param_i$ along with ID_i. This concludes the registration process for ID_i and the process is repeated for all authorized devices in the network. In this way, the verifier stores a list of IDs of all authorized IoT devices, their corresponding algorithm parameters, and a list of randomly generated nonces L_R. Note that L_R is also stored in the IoT device.

B. Authentication phase

Figure 4 shows the authentication phase. During this phase, the verifier collects one SRAM trace (S_T) from a prover (ID_A) and forwards it to the trained GN_A network for reconstruction. The RE is evaluated and compared to T_A. If the resultant RE lies below T_A, the prover is verified as ID_A (binary 1), else it is labeled as unauthorized (binary 0). The

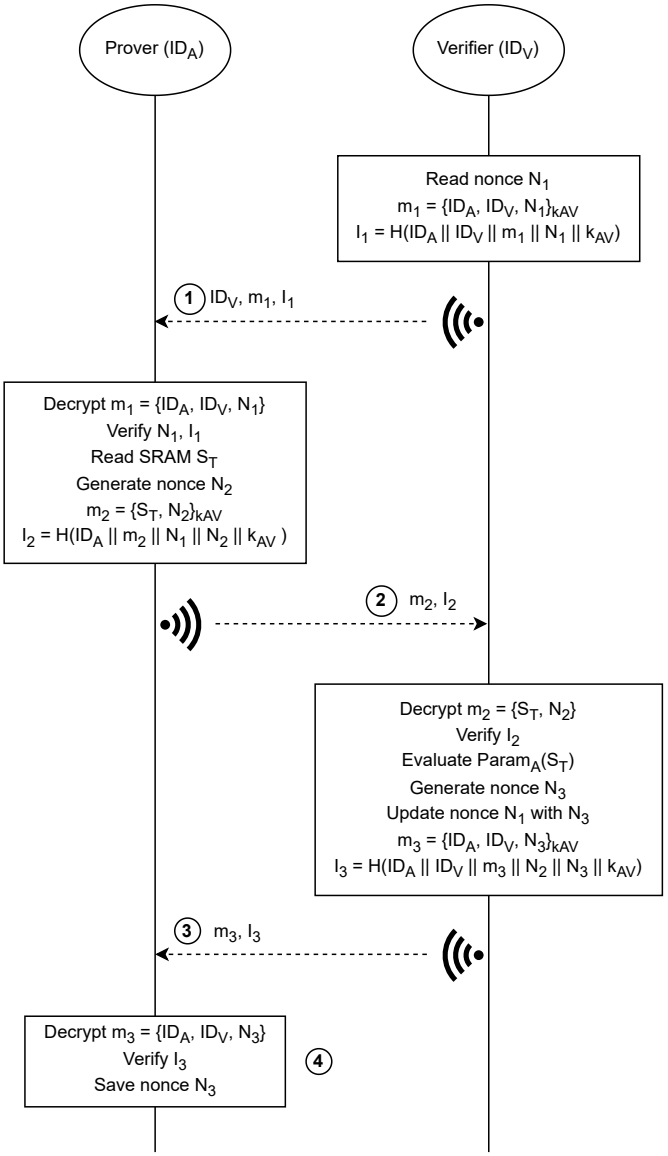


Fig. 5: The proposed fingerprinting protocol.

verifier fingerprints the prover by evaluating $Param_A(S_T)$ using the equation below.

$$Param(S_T) = \begin{cases} 0, & RE(S_T) > T_A \\ 1, & otherwise \end{cases}, \tag{4}$$

where RE is evaluated using Equation 1 and T_A is the anomaly threshold of the authorized device ID_A. The authentication phase is deployed in Step-2 of the proposed protocol, discussed in the next section.

V. Proposed Protocol

To identify compromised nodes in the IoT, the verifier periodically authenticates the IoT devices based on the fingerprinting algorithm. Figure 5 shows the proposed protocol. To verify the identity of a prover, the verifier must initiate one run of the protocol which involves the following four steps.

1) The verifier (ID_V) first attempts to establish a secure connection with a previously registered device in the network, say ID_A, and sends an SRAM fingerprint request. The first connection to an IoT device happens using one of the reserved nonces shared by the device and the verifier, represented by N_1. Subsequently, the nonce generated in step 3 is used. The verifier encrypts ID_A, ID_V, and N_1 into message m_1 using their shared secret key k_{AV} and creates an authentication parameter $I_1 = H(ID_A||ID_V||N1||k_{AS})$. ID_V, m_1 and I_1 are then sent to the prover.
2) The prover uses ID_V to select the corresponding key k_{AV} for decrypting m_1. It then uses the decrypted information to verify I_1 and N_1 where N_1 is matched with the list of reserved nonces stored in the IoT device. If successful, it reads one SRAM trace S_T, generates a random nonce N_2, and encrypts them into a message m_2. It also creates the authentication parameter I_2 and sends m_2 and I_2 as an authentication response to the verifier. Since the verifier is assumed to be a trusted device, I_1 is always authentic. However, if that assumption is not made and the verification of I_1 is unsuccessful, the prover identifies the supposed verifier as a malicious entity and blocks future connection requests from it.
3) The verifier decrypts message m_2 to retrieve S_T and N_2, and uses the decrypted data to verify I_2. If successful, it evaluates S_T on the algorithm parameters, $Param_A$, to obtain a fingerprint decision as per Equation 4. If the result is 1, the verifier has successfully attested the prover and it sends an encrypted message m_3 and hash I_3 as acknowledgment. It also updates N_1 for the next connection with N_3. However, if $Param_A(S_T)$ returns 0 or the verification of I_2 is unsuccessful, the protocol deletes $Param_A$ from the verifier memory and adds it to the list of untrusted devices, blocking all future connection requests from this device. In this way, the unauthorized device is isolated and does not impact the network.
4) The prover decrypts m_3 and verifies I_3. If successful, it updates the nonce N_1 with N_3 for the next connection with the verifier.

VI. Experiment Design

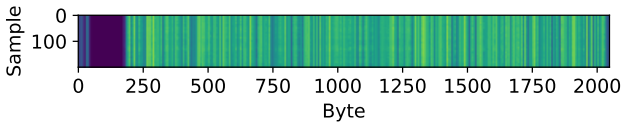
This section details the experimental setup of the paper.

A. Prover Devices

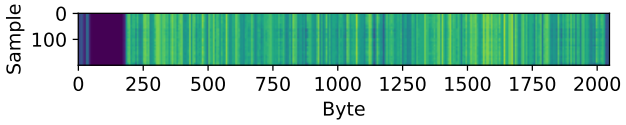
Three UNO Rev3 devices: Uno_1, Uno_2 and Uno_3 were used as provers in this paper, of which two devices (Uno_1 and Uno_2) were genuine Arduino products, while the third (Uno_3) was manufactured by Elegoo. The provers were loaded with an Arduino IDE file containing the necessary code to retrieve their 2kB SRAM traces. This setup helps us test the fingerprinting algorithm on devices of identical architecture manufactured by the same as well as different manufacturers.

B. Verifier Device and Simulation Environment

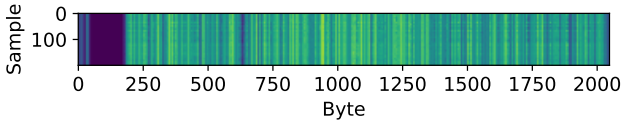
The verifier in this paper was a Dell Inspiron laptop with a 3 GHz 11th Gen Intel i7 vPRO processor and a 16 GB DDR4



(a) Uno_1



(b) Uno_2



(c) Uno_3

Fig. 6: The 2kB SRAM fingerprint dataset.

DRAM. The fingerprinting algorithm and data collection programs were written in Jupyter Notebooks and run on the laptop CPU. The main python libraries used in this paper were keras-2.9.0, numpy-1.21.5, pandas-1.4.2, matplotlib-3.5.1, and pyserial-3.3.

C. Dataset and Preprocessing

Figure 6 shows the 2kB SRAM fingerprint dataset collected for the three devices. 200 initial power-up SRAM states were collected manually for each prover by resetting the power to the devices. Looking at the memory usage results of the Arduino IDE, it was seen that the .data section occupied 184 bytes for all provers. The remaining 1864 bytes corresponding to the stack were selected for fingerprinting and divided by a factor of 255 to normalize the values to a range of [0,1]. The samples were randomized and the first 175 samples were used as training data while the remaining 25 were used for testing. Furthermore, 10 additional samples from Uno_1 were collected at different times and under different physical conditions to verify the robustness of the approach against varying environmental factors.

D. Generative Network Architectures

Three identical multi-layer perceptron networks were used as the device-specific GNs (GN_1, GN_2, and GN_3). The selected architecture is simplistic and comprises an input layer of 1864 normalized input bytes, one hidden layer of 20 neurons, and an output layer of 1864 neurons for the reconstructed bytes. All layers are activated using the ReLU function. Additionally, a dropout of 0.2 is introduced for the hidden layer. Although Stochastic Gradient Descent (SGD) has been shown to perform better than the Adam optimizer [18], the latter performs better in this experiment and is the optimizer of choice. The models were trained using MSE for 200 epochs at a learning rate of 0.001. Figure 7 shows the learning curves for the three GNs on the 2kB SRAM fingerprint dataset shown in Figure 6.

TABLE II: Prover authorization (%) using the three sets of algorithm parameters.

S.No.	Algorithm parameters	Registered device	Prover fingerprint (%)		
			Uno_1	Uno_2	Uno_3
1	$Param_1$	Uno_1	100	0	0
2	$Param_2$	Uno_2	0	100	0
3	$Param_3$	Uno_3	0	0	100

TABLE III: Execution time analysis on Jupyter notebook and the Intel i7 vPRO processor laptop.

Trace retrieval and processing (s)	Evaluation (s)		
	$Param_1$	$Param_2$	$Param_3$
2.1	0.0125	0.014	0.014

E. Anomaly Threshold Selection

Using Equation 3 on the train and test samples of the three devices, 0.0223, 0.0155, and 0.0124 were selected as the anomaly thresholds T_1, T_2, and T_3, respectively.

F. Algorithm Parameters

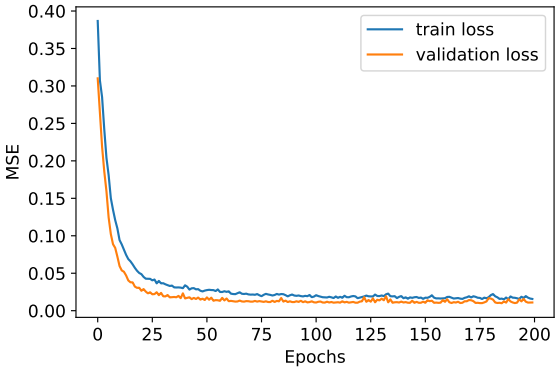
The obtained GNs (GN_1, GN_2, and GN_3) and anomaly thresholds (T_1, T_2, and T_3) were stored on the verifier as three sets of algorithm parameters ($Param_1$, $Param_2$, and $Param_3$).

VII. Results

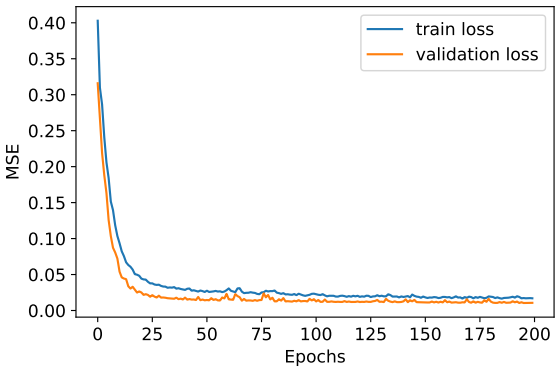
Figure 8 shows the detection box plots of the three provers on their respective algorithm parameters. The figure can be interpreted as follows: In Figure 8a, samples from Uno_1 lie below its anomaly threshold (T_1 = 0.0118), while those from Uno_2 and Uno_3 lie above it. This means that $Param_1$ is able to fingerprint Uno_1 and detects Uno_2 and Uno_3 as unknown devices. A similar inference can be made from Figures 8b and 8c. Table II shows that the proposed fingerprinting technique can identify authorized devices with 100% accuracy. Additionally, a 0% detection means the parameters detect a device different from their registered device. Based on the results, it can be inferred that one test sample is sufficient to evaluate each device, i.e., one run of the authentication protocol.

Table III shows the execution time for the fingerprinting process. The data sampling and preprocessing take 2.1 seconds per sample, while the prediction and error evaluation takes 0.0125 to 0.014 seconds. Since the proposed algorithm uses simplistic MLP networks and simple operations, it can also be deployed on low-power verifiers compatible with machine learning.

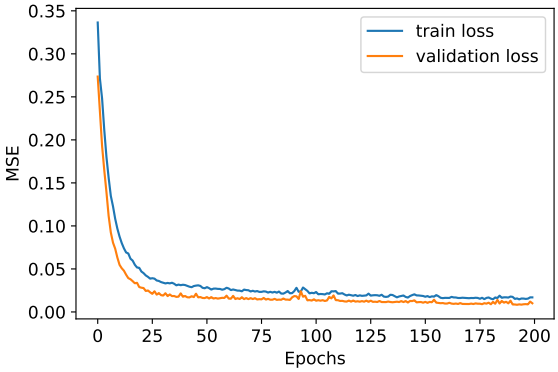
The obtained results validate the intended functionality of the proposed algorithm. Furthermore, the device-specific algorithm parameters can differentiate between the registered and unknown devices without the need to train on data from the latter. This means that the proposed method is self-sufficient and can perform a ”one vs any” classification.



(a) GN_1

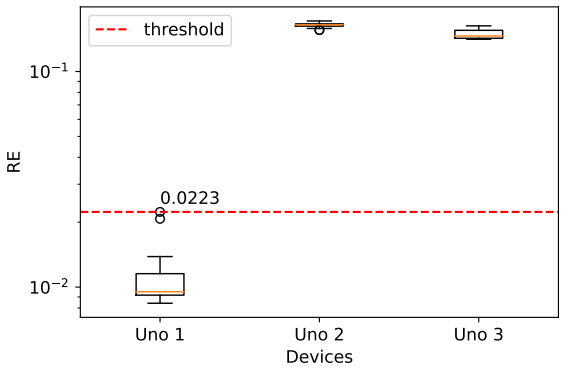


(b) GN_2

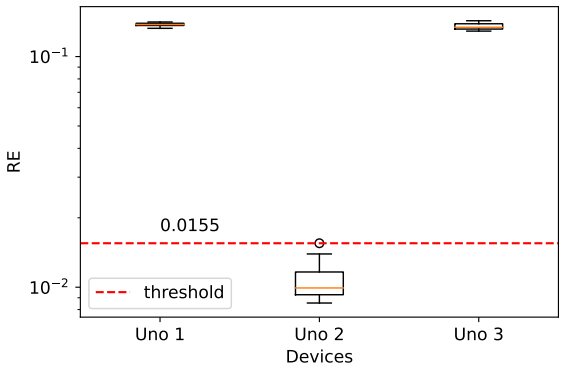


(c) GN_3

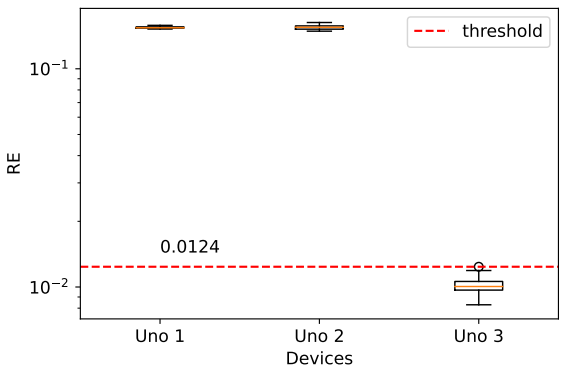
Fig. 7: Train and validation loss for the three generative networks.



(a) $Param_1$



(b) $Param_2$



(c) $Param_3$

Fig. 8: Fingerprint boxplots of test samples using the three algorithm parameter sets.

VIII. Security Analysis

Lemma 1. *The behavior of the SRAM cannot be predicted.*

Proof. Analysis of the pseudo-random SRAM states of the stack shows a list of possible values for each byte which are distinct for each prover. Assuming that every combination of unique byte values across all bytes is possible, the probability (P) of predicting the entire 1864-byte stack correctly can be calculated using the equation

$$P = \prod_{i=1}^{1864} P_i \tag{5}$$

where P_i is the probability of correctly predicting byte i. Based on our experiments, this value has a lower bound of 10^{-3800}. Factoring in the loose assumption of exhaustive combinations which is generous towards the attackers, the actual probabilities are expected to be even lower. Thus, the SRAM trace is unique to each device and the adversary can make a random guess to predict the state of a stack.

Therefore, the advantage of an adversary is bounded by $\text{Adv}_{\mathcal{A}}^{SRAM} \leq \frac{1}{10^{-3800}}$. Note that for practical reasons, the number of stack bytes used for fingerprinting may be reduced based on the required security level.

The discussion in Section II and results from Section VII support the universality of this lemma. □

Theorem 2. *Authentication: If a verifier completes one run of the protocol, it has indeed done so with a legitimate device.*

Proof. An adversary may attempt a proxy attack. In this type of attack, an adversary uses an identical IoT device to impersonate an authorized IoT device. To successfully launch a proxy attack, an adversary needs to send an authentic SRAM trace to the verifier in response to an authentication request. However, by Lemma 1 this is not possible. Thus, it is not possible for an adversary to authenticate itself as a legitimate IoT device. □

Theorem 3. *Availability: An IoT device registered with the verifier is always available.*

Proof. An adversary may launch a Denial of Service (DoS) attack by impersonating as a legitimate verifier [19] and sending authentication requests more frequently. To successfully launch this type of attack the adversary must generate a valid authentication parameter $I_1 = H(ID_A||ID_V||m_1||N_1||k_{AV})$. However, this is not possible without knowledge of the secret key k_{AV} shared between the prover and the verifier only.

An adversary may also attempt at replaying a previously sent authentication request. To successfully launch this type of attack the adversary needs to generate a valid authentication parameter I_1. However, this is not possible without knowledge of the nonce N_1, which can not be revealed without knowledge of the secret key K_{AV}.

The adversary may also affect the availability of an IoT device by dropping message 2 sent by the prover to the verifier (carrying the authentication response) or message 3 sent by the verifier (carrying the new value for the initiating random nonce, i.e., N_3) to the prover. This type of attack is called a de-synchronization attack. However, the verifier maintains an extra list of random nonces, L_R, as described in Section IV, which it can use to re-synchronize with an IoT device.

Thus, the proposed protocol is secure against DoS attacks and ensures availability of the IoT devices. □

IX. Conclusion

This paper proposed a lightweight, intelligent algorithm for hardware fingerprinting and the corresponding authentication protocol using initial power-up SRAM states as a physical fingerprint to manage trust in IoT networks. SRAM samples from three Arduino UNO provers were used to train their respective generative networks and evaluate anomaly thresholds. The resulting algorithm parameters were tested on data from all three devices. The results showed the ability of the proposed method to fingerprint registered devices and detect unauthorized ones with 100% accuracy. Execution-time analysis of the proposed results showed the practicality of the proposed method, with 2.1 seconds needed to collect and preprocess a trace and 0.0125 to 0.014 seconds for evaluation. To maintain trust in the IoT network, the verifier initiates the protocol with each IoT device periodically. Based on the outcome, devices may be flagged as compromised. The brute force attack probability was found to be of the order 10^{-3800} and a security analysis of the proposed protocol shows that it is secure against various types of attacks.

References

[1] G. M. Honti and J. Abonyi, "A review of semantic sensor technologies in internet of things architectures," *Complexity*, vol. 2019, 2019.

[2] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li, "A survey on edge and edge-cloud computing assisted cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7806–7819, 2021.

[3] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5g and internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.

[4] L. Chettri and R. Bera, "A comprehensive survey on internet of things (iot) toward 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2019.

[5] S. Kumar, P. Tiwari, and M. Zymbler, "Internet of things is a revolutionary approach for future technology enhancement: a review," *Journal of Big data*, vol. 6, no. 1, pp. 1–21, 2019.

[6] 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.

[7] K. Yang, Q. Li, and L. Sun, "Towards automatic fingerprinting of iot devices in the cyberspace," *Computer networks*, vol. 148, pp. 318–327, 2019.

[8] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "Deft: A distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 940–952, 2018.

[9] U. Javaid, M. N. Aman, and B. Sikdar, "Defining trust in iot environments via distributed remote attestation using blockchain," in *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2020, pp. 321–326.

[10] M. N. Aman, M. H. Basheer, and B. Sikdar, "A lightweight protocol for secure data provenance in the internet of things using wireless fingerprints," *IEEE Systems Journal*, vol. 15, no. 2, pp. 2948–2958, 2021.

[11] M. N. Aman, K. C. Chua, and B. Sikdar, "Mutual authentication in iot systems using physical unclonable functions," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1327–1340, 2017.

[12] D. E. Holcomb, W. P. Burleson, K. Fu *et al.*, "Initial sram state as a fingerprint and source of true random numbers for rfid tags," in *Proceedings of the Conference on RFID Security*, vol. 7, no. 2, 2007, p. 01.

[13] K. Merchant, S. Revay, G. Stantchev, and B. Nousain, "Deep learning for rf device fingerprinting in cognitive communication networks," *IEEE journal of selected topics in signal processing*, vol. 12, no. 1, pp. 160–167, 2018.

[14] V. Kohli, A. Chougule, V. Chamola, and F. R. Yu, "Mbre ids: An ai and edge computing empowered framework for securing intelligent transportation systems," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.

[15] A. Chougule, V. Kohli, V. Chamola, and F. R. Yu, "Multibranch reconstruction error (mbre) intrusion detection architecture for intelligent edge-based policing in vehicular ad-hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[17] M. Sabuhi, M. Zhou, C.-P. Bezemer, and P. Musilek, "Applications of generative adversarial networks in anomaly detection: a systematic literature review," *IEEE Access*, vol. 9, pp. 161 003–161 029, 2021.

[18] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628*, 2017.

[19] 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.