Alexis: Anomaly-Based Intrusion Detection in FPGA-Enabled Cyber-Physical Systems

Umara Hanif, Muhammad Naveed Aman, *Senior Member, IEEE*, and Biplab Sikdar, *Fellow, IEEE*

***Abstract*—Ensuring the security of FPGA-based systems is crucial, especially in cyber-physical environments where reliability and security are paramount. This paper introduces *Alexis*, a novel anomaly detection framework leveraging autoencoder neural networks to identify hardware Trojans in FPGA bitstreams. *Alexis* meticulously examines bitstream data for discrepancies, thereby providing a robust approach to mitigate security breaches. Experimental evaluations highlight the superiority of *Alexis*, achieving an accuracy upto 91%, significantly outperforming traditional approaches. Moreover, the Mean Squared Error (MSE) and Mean Absolute Error (MAE) values consistently indicate minimal reconstruction errors, thereby demonstrating the reliability of the autoencoder's learning capabilities. The computational efficiency of *Alexis*, with an average time overhead of less than 0.021 seconds and memory usage around 1.14 MiB, underscores its suitability for resource-constrained environments. Additionally, the high Area Under the Curve (AUC) values range upto 0.96, illustrate the model's exceptional performance in accurately distinguishing anomalies from normal bitstreams. These results establish *Alexis* as a scalable and efficient solution for enhancing the security of FPGA-enabled cyber-physical systems.**

***Index Terms*—Cyberphysical systems security, IoT, field programmable gate arrays, hardware trojans, anomaly detection.**

Nomenclature

Abbreviations and Acronyms

Vf	We denote the verifier with.
Pf	Prover FPGA with.
Rg	Golden Reference with.
A	Adversary with.
C	Challenger with.

I. Introduction

ANOMALY detection is a cornerstone in ensuring the security and reliability of diverse systems, spanning hardware and software domains, particularly vital in the context of cyber-physical systems. Its essence lies in pinpointing patterns or events that deviate from anticipated behaviors or normal operations. In hardware systems, anomalies may stem from a myriad of factors, including design errors, manufacturing defects, and malicious attacks, which, if left unaddressed, can culminate in system failures, downtime, and security breaches [6].

The escalating complexity and scale of contemporary hardware systems, especially in the case of Field-Programmable Gate Arrays (FPGAs), present a formidable challenge for anomaly detection. FPGAs find significant use in critical applications such as aerospace, defense, and cyber-physical systems, where security and reliability are paramount [33] [34] [35] [37] [38]. The intricate design of FPGAs involves integrating a multitude of programmable logic blocks, memory elements, and interconnects, rendering the detection of anomalies a daunting task due to the sheer scale and complexity of the system. Notably, hardware Trojans, insidious modifications to electronic hardware designs, pose a significant security threat by exploiting vulnerabilities and potentially causing system failures or data leaks [16] [18].

To counteract these challenges, various machine learning-based approaches have been proposed, with autoencoders emerging as a potent tool. Autoencoders, a class of neural networks, excel in learning to reconstruct input data by compressing it into a lower-dimensional representation. Demonstrating effectiveness across diverse data types, including images, time series, and sensor data, autoencoders find applications in anomaly detection [9]. This paper harnesses autoencoders to detect anomalies [39] [40] in bitstream data generated during the FPGA configuration process, a critical aspect for the security of cyber-physical systems.

FPGA attestation, a process verifying the authenticity and integrity of FPGA designs or bitstreams, plays a pivotal role in ensuring the security of FPGA-based systems. It involves confirming that an FPGA remains untampered and that the loaded bitstream aligns with the expected design [2]. FPGA attestation proves invaluable in detecting hardware Trojans, which could compromise the security and reliability of the system [16] [22]. This paper presents an autoencoder-based approach for detecting anomalies in FPGA bitstream data during the configuration process. The method involves representation learning through an autoencoder to measure reconstruction errors, which serve as indicators for anomalies. Specifically, this approach compares specific entries in the bitstream against a Golden Reference (Rg), a trusted and verified baseline representing an authentic FPGA bitstream free from hardware Trojans or anomalies [6]. Any

Received 11 October 2024; revised 29 October 2025; accepted 13 December 2025. Date of publication 29 December 2025; date of current version 5 February 2026. This work was supported in part by A*STAR, CISCO Systems (USA) Pte. Ltd and in part by the National University of Singapore under its Cisco-NUS Accelerated Digital Economy Corporate Laboratory under Award I21001E0002. *(Corresponding author: Muhammad Naveed Aman.)*

Umara Hanif and Biplab Sikdar are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (e-mail: umara.hanif@u.nus.edu; bsikdar@nus.edu.sg).

Muhammad Naveed Aman is with the School of Computing, University of Nebraska-Lincoln, Lincoln, NE 68588 USA (e-mail: Naveed.aman@unl.edu).

deviations identified by the autoencoder are flagged as potential anomalies. The effectiveness of this approach is validated using real bitstream data, demonstrating high accuracy and its potential to enhance the security of cyber-physical systems.

The Golden Reference (Rg) is derived from a rigorously tested and validated FPGA configuration, stored in a secure repository. It remains static over time to ensure consistency in anomaly detection, serving as a stable benchmark against which other configurations are compared [26]. For FPGA security, it is crucial to protect configuration integrity and guard against hardware Trojans. The proposed methodo, Alexis, offers robust Trojan detection, ensuring that only trusted designs are executed [16]. Alexis minimizes false positives, secures bitstream storage, and adapts to evolving security threats. Tested on real-world FPGA devices, Alexis effectively balances accuracy and performance while leaving room for future enhancements, such as incorporating additional security layers [18] [29]. This methodology provides a strong foundation for securing FPGA-based systems, particularly in critical cyber-physical systems.

The **major contributions** of this paper are as follows:

1) A bitstream-based anomaly detection technique to attest the integrity of FPGAs and detect Altered functionality, Fault-inducing, Configuration, and Information leakage hardware Trojans.
2) Experimental evaluation of the proposed attestation technique on actual hardware.

The rest of the paper is organized as follows. Section II provides a literature review of related work. Section III describes the system and attack model, and Section IV describes the proposed autoencoder-based approach in detail, including the training procedure, and anomaly detection process. Section V presents a security analysis, Section VI presents the experimental setup, and Section VII describes the experimental results of the proposed approach. Finally, Section VIII concludes the paper and provides possible future research directions.

II. Related Work

The use of hardware security mechanisms, such as attestation, has become increasingly important in recent years, particularly for embedded devices and the Internet of Things (IoT) based systems. Attestation allows devices to prove their integrity to other parties and is a critical component for securing the communication and interactions between devices in a network.

Several approaches to attestation have been proposed in the literature. HAtt (Hybrid Remote Attestation for Internet of Things with High Availability) [1] is a hybrid approach that combines both hardware-based and software-based attestation to achieve high availability and robustness against network failures. A novel FPGA Architecture and Protocol for Self-Attestation of configurable hardware is presented in [2]. The authors propose a hardware-based self-attestation mechanism for FPGAs, which can be used to verify the integrity of an FPGAs' configuration bitstream.

Software-based attestation approaches have also been developed, such as SWATT (Software-based attestation for embedded devices) [3]. SWATT uses software-based measurements of the device's execution environment to verify its integrity. Machine learning techniques have also been used for attestation purposes [4]. This approach uses machine learning algorithms to analyze memory traces from devices to detect anomalies and verify their identity, but its practicality and resilience under real-world adversarial scenarios warrant deeper investigation.

In SHeLA [5] (Scalable Heterogeneous Layered Attestation), the authors developed a swarm remote attestation technique where classical remote attestation protocol works with one *Pf* and one *Vf*. This technique faces the challenges of scalability of swarm and can't prevent local adversaries from carrying out eavesdropping or snooping attacks. Also, it doesn't prevent any hardware attacks.

FPGA-Based Remote Code [6] has been proposed to use reconfigurable computing to build consistent architecture for conducting attestation. This technique is useful for the analysis of replay attacks, cryptanalysis attacks, MITM, and DDoS. Together with obfuscation, remote-dynamic update of FPGA gives attackers a limited time to succeed in their actions.

Reactive attestation [7] is an anti-tampering approach that automatically applies to the target program by using remote attestation for detection. This technique uses client/server code splitting, If the client provides evidence of its integrity then the part gets executed on the server, otherwise server stops serving. Local reactions adopt patterns that are recognizable with static and dynamic analysis and use challenge-response mechanisms to avoid replay attacks. Time-based attestation is vulnerable to proxy attacks and relies on trusted information about precise client hardware configuration and reactive attestation can be detected and defeated by automatic tools/expert hackers.

POSTER (practical embedded remote attestation using physically unclonable functions) [8] is the remote attestation using PUFs. It provides security against collision attacks, allows authentication of remote *Pf*s, and enables detection of hardware attacks on *Pf*. Overall, while the use of PUFs in remote attestation is intriguing, a more thorough analysis and evaluation of the approach's robustness and feasibility are needed to strengthen the POSTER's impact and relevance. Representation learning-based anomaly detection is another approach that has gained popularity in recent years. A practical autoencoder-based anomaly detection method was proposed in [9], which uses vector reconstruction error to detect anomalies in data streams. This approach has been successfully applied to various domains, including industrial control systems. Autoencoders have been successfully applied in various security domains, including network intrusion detection, fraud detection, and malware detection, demonstrating their ability to identify subtle deviations in data [41] [42]. Fuzzing hardware-like software [10] is an emerging area of research that focuses on applying software fuzzing techniques to hardware systems, such as FPGAs, to detect and exploit vulnerabilities.

In the cyber-physical systems (CPS), recent literature highlights both advancements and areas needing further exploration. In "Artificial Intelligence in Smart Logistics Cyber-Physical Systems: State-of-The-Arts and Potential Applications" [29] Yang Liu et.al, delve into AI's role in optimizing smart logistics, underscoring its potential to revolutionize efficiency and

TABLE I
COMPARISON OF EXISTING ATTESTATION TECHNIQUES WITH ALEXIS

Technique	SWATT [3]	Fuzzing HW like SW [10]	SACHa [2]	Alexis
Key Features	Memory attestation, no hardware usage	Hardware Vulnerabilities, Test Generation, Input Mutation	FPGA Self-attestation, Configurable Hardware	Anomaly detection in bitstream data
Approach	Checksum-based memory verification	Directed Fuzzing, Code Coverage, Mutation Testing	Protocol-based, Cryptographic checksum computation	Autoencoder-based anomaly detection using PyTorch
Training Data	Not applicable, verification procedure for runtime memory assessment	Device Firmware, Hardware Simulation Binaries	Trusted Configurations Memory, FPGA Frames	Synthetic and real-world bitstream data
Anomaly Detection	Detects memory content changes	Code Paths Exploration	Deviation Detection, Runtime Monitoring	Maximum error threshold-based detection
Evaluation Metrics	Detection rate, false positives	Code Coverage, Crash Detection	Integrity Validation, Configuration Verification	Accuracy, MSE, MAE
Advantages	No secure hardware required	Bus-centric harness, Automation, Scalability	Configurable Trust, Real-time Verification	Effective anomaly detection, robustness against Trojans, training flexibility
Limitations	Temporal vulnerabilities, no virtual memory support	No notion of hardware sanitizer and RTL hardware	Protocol Overhead, Complexity Challenges	Requires labeled data for training

decision-making processes. Despite its thorough review, the paper overlooks the need for a comprehensive AI integration framework and minimally addresses the socio-technical and interoperability challenges posed by AI implementation in logistics.

"Monitoring and Defense of Industrial Cyber-Physical Systems Under Typical Attacks: From a Systems and Control Perspective" [30] examines strategies for protecting industrial CPS against cyber threats, focusing on systems and control approaches. Although it outlines various defense mechanisms, the paper could further benefit from discussing adaptive strategies that preempt future threats and more seamless integration of these mechanisms within existing cybersecurity frameworks.

Lastly, in "Robust and Resilient Distributed MPC for Cyber-Physical Systems Against DoS Attacks" [31] the authors propose a framework to enhance the resilience of distributed MPC systems against DoS attacks. While presenting a solid foundation for countering such cyber threats, the paper's limited testing scenarios and lack of discussion on scalability and applicability to other cyber threats suggest room for broader empirical research.

Together, these works contribute valuable insights into the advancement of CPS security and logistics efficiency through AI and control systems. However, they collectively highlight the necessity for more integrated frameworks, adaptive defense strategies, and comprehensive empirical validations to tackle the evolving landscape of cyber threats and technological integrations in CPS.

Alexis is a promising approach that can be applied to various domains, including IoT systems, for attestation and to ensure the integrity of the devices. Table I compares the proposed technique, Alexis, with the most relevant previous techniques based on different features.

III. SYSTEM AND THREAT MODEL

A. System Model

The system model is shown in Fig. 1 with N Provers (*Pfs*) and a single Verifier (*Vf*), the objective is to verify the Pfs based on bitstream data using an autoencoder.

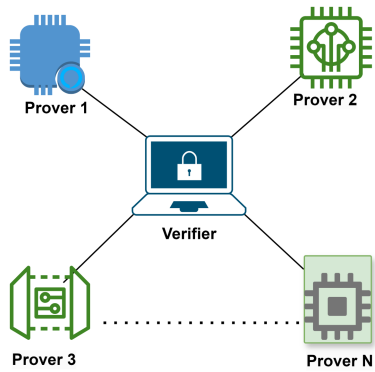


Fig. 1. System model.

Pfs: These are the entities or devices whose integrity and authenticity need to be verified. Each Pf generates and presents a bitstream data set for verification. The number of *Pf*s can vary from one (a single *Pf*) to N (multiple *Pf*s).

Vf: The *Vf* is a trusted entity responsible for verifying the integrity and authenticity of the *Pf*s.

B. Assumptions

Data Integrity: The *Pf*s' bitstream data is assumed to be free from unintentional errors or corruptions that could affect its functionality. The *Vf* expects the data to be accurate.

Golden Reference Availability: The proposed method assumes access to a verified and validated "Golden Reference" (*Rg*), which serves as a trusted baseline for comparison with other FPGA bitstreams. The assumption of a trusted Golden Reference is critical to the success of Alexis. In environments where such a reference is not available, alternative approaches such as consensus-based validation across multiple devices could be explored. These methods would involve cross-referencing the integrity of bitstreams across a network of devices to establish a baseline reference dynamically [5].

Integrity of Training Data: The method assumes that the training data used for the autoencoder model is reliable and free from tampering or manipulation.

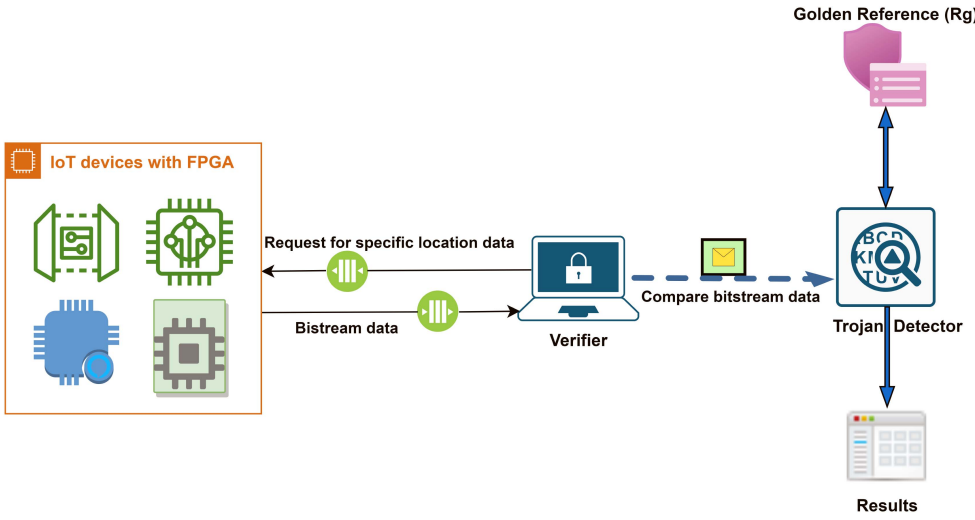


Fig. 2. Block diagram of Alexis.

Adversarial Knowledge: The method assumes that adversaries may have knowledge of the presence of the defense mechanism and may attempt to bypass it.

FPGA Hardware: The method is designed for use with Field-Programmable Gate Arrays (FPGAs) and assumes the presence of such devices.

C. Threat Model

Assets: Integrity of the FPGA configuration (bitstream), provenance of the golden reference, and verifier keys/channels.

Adversary Goals: (i) Inject a Trojan by modifying the bitstream; (ii) supply a different bitstream while presenting a benign one to the verifier (relay/fake-report); (iii) *evade* Alexis by crafting inputs close to the learned manifold.

Capabilities: White-box knowledge of features and model parameters; access to the supply chain or deployment step where bitstreams are provisioned; ability to control what is sent to the verifier; *no* ability to break standard cryptographic collision resistance or compromise the verifier.

Assumptions: A secure verifier–prover channel exists; training data for Alexis is clean; the bitstream format exposes stable structural features (e.g., frames/words) after toolchain processing.

Non-goals: Alexis does not detect runtime-only malicious behavior when the configured bitstream equals a trusted golden; nor does it prevent a device that *relays* a valid golden bitstream to the verifier while operating maliciously. Such cases require runtime attestation and a hardware root of trust.

Objective: Under these conditions, Alexis flags configuration-time anomalies that are *consistent* with adversarial tampering or unexpected toolflow drift, complementing cryptographic checks rather than replacing them.

IV. Proposed Methodology

An overview of the proposed attestation technique is shown in Fig. 2. The details of the proposed algorithm to detect the presence of hardware Trojans/Anomalies in FPGA devices by analyzing the bitstream file that configures the FPGA are given in Algorithm 1. "Hashed values" refer to the output of a hash function applied to selected elements of the FPGA bitstream. A hash function maps an input segment of the bitstream to a fixed-size digest; even slight input changes produce a different digest, which we exploit for integrity checks. The methodology uses these hashed values as compact surrogates of relevant bitstream characteristics, enabling lightweight comparisons for anomaly detection without operating on the full file.

A. Bitstream Feature Selection

Instead of hard-coding a fixed suffix, we select the top-K high-variability segments using a simple, device-agnostic criterion. Let $\{b_i\}$ denote line- or frame-level tokens extracted from the bitstream. For each position i, compute normalized Shannon entropy H_i across clean bitstreams. We use the indices $\mathcal{I}_K = \text{argsort}_i(H_i)$ (top-K) as features for Alexis. In our datasets, many high-entropy segments *happen* to concentrate near the end due to toolflow/device specifics, but this is not universal; the entropy-ranked selector adapts across placements and designs while keeping feature dimensionality fixed.

The proposed method has the following steps:

1) Read the bitstream file of the FPGA device.
2) Initialize a container for hashed values and the selected indices $\mathcal{I}_K$.
3) For each index $i \in \mathcal{I}_K$, extract the corresponding bitstream segment and compute its hash.
4) Store the hashed values in order of i.
5) Compute a checksum over the K hashed values using Algorithm 2.
6) Input the checksum to the Trojan detector to decide whether a hardware Trojan/anomaly is present.
7) If an anomaly is detected, use the index set $\mathcal{I}_K$ and the hashed-value container to localize the affected region(s).
8) Output the detection result, including location cues when applicable.

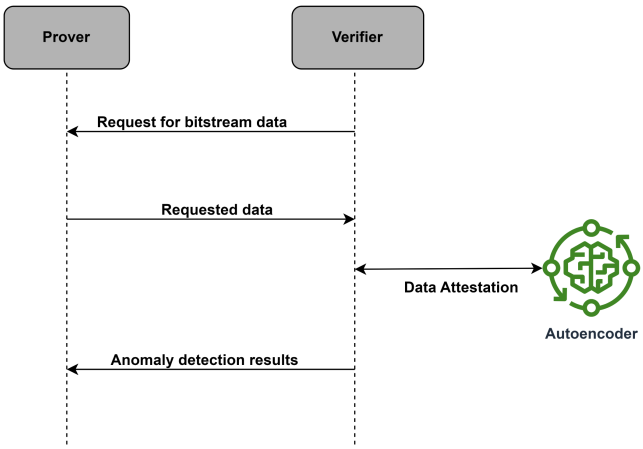


Fig. 3. Overall operation.

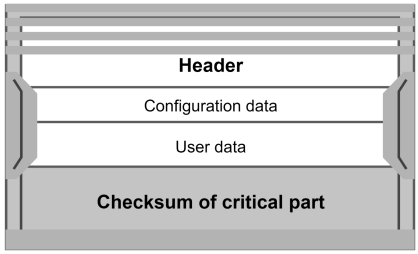


Fig. 4. Bitstream Feature Selection: entropy-ranked top-K segments are used for hashing and attestation.

Algorithm 1: Proposed Attestation Algorithm.

$bitstream \leftarrow read\_bitstream(FPGA\_device)$
$\mathcal{I}_K \leftarrow select\_topK\_indices()$ {entropy-ranked indices from clean fleet}
$hashed \leftarrow [\,]$
for i in $\mathcal{I}_K$ **do**
  $seg \leftarrow extract(bitstream,\ i)$
  $h \leftarrow hash(seg)$
  append$(hashed,\ h)$
end for
$checksum \leftarrow calculate\_checksum(hashed)$
if $Trojan\_detection\_model(checksum)$ **then**
  $print$(" $Anomaly\ detected\ at\ indices:$ ", $\mathcal{I}_K$)
else
  $print$(" $No\ anomaly\ detected\ in\ FPGA\ device$ ")
end if

Fig. 3 shows the overall operation of the system: a *Vf* sends an attestation request to a *Pf* (the FPGA device), which returns the requested bitstream data; the *Vf* runs the proposed technique and reports whether the device passes attestation.

The process starts by reading the bitstream file and computing hash values over the entropy-ranked top-K high-variability segments selected by $\mathcal{I}_K$, as shown in Fig. 4. The hash values are stored in an array and a checksum is computed for this array (Fig. 5). The checksum serves as a compact identifier of the

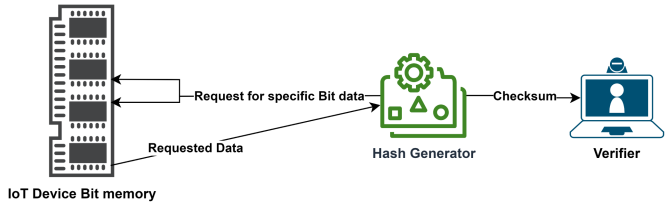


Fig. 5. Checksum computation.

Algorithm 2: Checksum Computation Algorithm.

Require: $hashed\_values$: List of hashed values
$checksum \leftarrow 0$ {Initialize the checksum to zero}
for $hashed\_value$ in $hashed\_values$ **do**
  {Iterate through the list of hashed values}
  $checksum \leftarrow checksum + hashed\_value$
  {Add the current hashed value to the checksum}
end for
return $checksum$ {Return the computed checksum}

selected hashes; any modification to the corresponding bitstream segments changes the checksum and is flagged by the detector.

We proposed Algorithm 2 to calculate a checksum from a list of hashed values. This checksum serves as a unique identifier for the provided hashed values and is a fundamental component of the proposed method for detecting anomalies in FPGA bitstreams. It computes a checksum, which is a numerical value that represents the aggregate of all hashed values in the given list using the following steps:

1) Initialize the checksum variable to 0. This is the variable that will store the computed checksum.
2) The algorithm iterates through each hashed value in the list of hashed values.
3) For each hashed value, it adds the hashed value to the checksum variable. This process continues until all hashed values are processed.
4) Finally, the algorithm returns the computed checksum.

A list containing individual hashed values, denoted as h_1, h_2, h_3,..., h_n, where n is the number of hashed values in the list.

$$\text{checksum} = \sum_{i=1}^{n} h_i \tag{1}$$

In this representation, Σ denotes the summation of elements. i is the index variable ranging from 1 to n. h_i represents each hashed value.

The proposed Trojan detector is then used to check whether the calculated checksum matches the expected checksum of the *Rg*. If the checksum does not match, then it is likely that a hardware Trojan has been added to the FPGA device.

The proposed Trojan detector uses an autoencoder, an unsupervised learning technique used for feature extraction and dimensionality reduction. It is composed of two parts, the encoder and the decoder. The encoder takes an input, compresses it into a lower-dimensional representation, and passes it to the decoder to reconstruct the original input. Auto-encoders can be trained

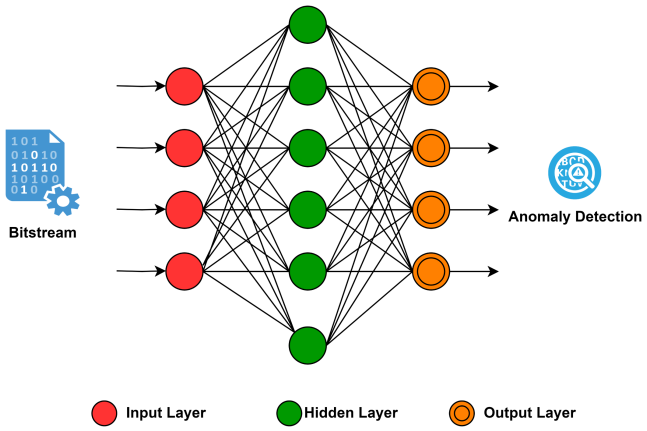


Fig. 6. Auto-encoder.

using various optimization algorithms, one of which is the Adam optimizer. The [32]Adam optimizer is an adaptive learning rate optimization algorithm that is widely used in deep learning. It adapts the learning rate for each parameter based on the first and second moments of the gradients. The optimizer computes an exponentially decaying average of the past gradients to update the learning rate which is adaptive for each parameter.

During training in the proposed method, an auto-encoder takes the input checksum data as shown in Fig. 6 and passes it through the encoder to obtain a lower-dimensional representation. The lower-dimensional representation is then passed through the decoder to reconstruct the original input. The difference between the reconstructed input and the original input is used to compute the reconstruction loss, which is minimized during training using the Adam optimizer. By minimizing the reconstruction loss, the auto-encoder learns to compress the input data into a lower-dimensional representation and reconstruct it accurately. Once the auto-encoder is trained, it can be used for anomaly detection by comparing the reconstruction loss of new data with the reconstruction loss of the training data. If the reconstruction loss is above a certain threshold, it indicates that the new data is anomalous. We used empirical data to detect this threshold.

The proposed attestation technique can be used by FPGA designers and users to verify the authenticity and integrity of the FPGA bitstream file and ensure that no hardware Trojans have been added to the FPGA device. This can enhance the security and reliability of IoT devices that use FPGAs, especially in critical applications such as aerospace, defense, and healthcare.

V. Security Analysis

The security analysis of Alexis highlights its robustness against adversarial threats, covering mathematical foundations, real-world applicability, and theoretical guarantees.

Checksum Computation and Integrity Verification: Alexis uses a SHA-256 checksum to ensure FPGA bitstream integrity. SHA-256, known for its collision resistance, verifies that the original bitstream X matches the golden reference X_g. The checksums are computed as follows:

$$h_X = \text{SHA-256}(X), \quad h_{X_g} = \text{SHA-256}(X_g).$$

An anomaly is detected if the checksums differ:

$$h_X \neq h_{X_g}.$$

Given the cryptographic strength of SHA-256, the probability of finding a collision, where $h_X = h_{X_g}$ for $X \neq X_g$, is negligible:

$$P(\text{Collision}) \approx \frac{1}{2^{128}}. \tag{2}$$

This high level of security ensures that any tampering with the bitstream, no matter how subtle, will be detected.

Lemma 1: The probability of an adversary successfully tampering with the bitstream such that the modified bitstream X' produces the same checksum as the golden reference X_g is negligible.

Proof: Assume that an adversary attempts to generate a bitstream X' such that:

$$\text{SHA-256}(X') = \text{SHA-256}(X_g).$$

Given the properties of SHA-256, the best strategy for the adversary is to attempt a brute-force attack to find a collision. The probability of a successful collision in SHA-256 is $\frac{1}{2^{128}}$, which is computationally infeasible. Therefore, the advantage of the adversary, $Adv^A_{SHA-256}$, is:

$$Adv^A_{SHA-256} \approx \frac{1}{2^{128}}. \tag{3}$$

This proves that the likelihood of undetected tampering is negligible.

Autoencoder-Based Anomaly Detection: The core of Alexis is an autoencoder neural network trained on golden reference bitstreams to learn the normal patterns and structures of FPGA configurations. The autoencoder is defined as:

$$AE : X \rightarrow \hat{X},$$

where X is the input bitstream and $\hat{X}$ is the reconstructed bitstream. The autoencoder minimizes the reconstruction error, typically measured by Mean Squared Error (MSE):

$$\text{MSE}(X, \hat{X}) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2, \tag{4}$$

where n is the number of data points in the bitstream.

Lemma 2: The reconstruction error $\text{MSE}(X, \hat{X})$ exceeds a predefined threshold ϵ, if the bitstream is infected.

Proof: The autoencoder is trained to minimize the reconstruction error for normal (untampered) bitstreams, such that:

$$\Theta_{AE} = \text{argmin}_\Theta \left(\sum_{i=1}^{n} \text{MSE}(X_i, \hat{X}_i; \Theta) \right), \tag{5}$$

where Θ_{AE} are the trained parameters of the autoencoder.

Given a threshold ϵ determined from the training data, if the reconstruction error for a new bitstream X' exceeds ϵ, i.e.,

$$\text{MSE}(X', \hat{X}') > \epsilon,$$

then X' is considered anomalous. The selection of ϵ is critical and is typically chosen based on the distribution of errors in the training data to minimize both false positives and false negatives.

Real-World Applicability, Case Study: In an industrial scenario, an attacker targets an FPGA-based control system by modifying the bitstream X to a tampered bitstream X'. Alexis detects this modification using two mechanisms:

1. *Checksum Verification:* The SHA-256 checksum of X' will differ from that of the golden reference X_g:

$$\text{SHA-256}(X') \neq \text{SHA-256}(X_g).$$

2. *Autoencoder Anomaly Detection:* The autoencoder, trained on untampered bitstreams, will produce a reconstruction $\hat{X}'$ for the tampered bitstream X' with a high reconstruction error:

$$\text{MSE}(X', \hat{X}') > \epsilon.$$

According to lemma 2, both of these mechanisms ensure that the tampered bitstream is flagged as anomalous, thereby preventing the deployment of the compromised FPGA in a critical system.

Theorem 3: If Alexis attests an FPGA device as Trojan-free, the probability P_T of a Trojan being present is negligibly small, with $P_T \propto Adv^A_{SHA-256} + P(\text{Autoencoder Bypass}$, ensuring high security.

Proof: Let Vf be the Alexis verification function, combining checksum verification and autoencoder-based anomaly detection. The verification is modeled as a game between a challenger C and adversary A, where A modifies the bitstream to X'. If C verifies $Vf(X') = 1$, the bitstream is considered Trojan-free. The probability of A successfully deceiving Vf is:

$$P(\text{Trojan Insertion}) \leq Adv^A_{SHA-256} + P(\text{Autoencoder Bypass}), \tag{6}$$

with

$$Adv^A_{SHA-256} \approx \frac{1}{2^{128}} P(\text{Autoencoder Bypass}) \approx \epsilon,$$

where ϵ is a small positive value (e.g., 10^{-9}), representing the adversary's success rate for bypassing detection in a well-trained model, ensuring Alexis' high security against Trojan injection.

Possibility of Mitigating Side-Channel Attacks: Alexis can mitigate side-channel attacks by detecting anomalies in FPGA power consumption patterns using an autoencoder trained on normal traces, flagging deviations as potential hardware Trojan attempts.

Lemma 4: If the power consumption P deviates beyond ϵ and the variance σ_P^2 of the reconstruction error exceeds ϵ_v, the probability P_{SC} of a side-channel attack or hardware Trojan increases, where $P_{SC} \propto \frac{\sigma_P^2}{\epsilon_v}$.

Proof: Let P represent the power consumption pattern of the FPGA during normal operation. The autoencoder is trained on these normal power patterns P and reconstructs them with minimal error. The reconstruction error is measured by the mean squared error (MSE):

$$\text{MSE}(P, \hat{P}) = \frac{1}{n} \sum_{i=1}^{n} (P_i - \hat{P}_i)^2, \tag{7}$$

where $\hat{P}$ is the reconstructed power consumption pattern from the autoencoder.

In addition to the MSE, the variance σ_P^2 of the reconstruction error is computed as:

$$\sigma_P^2 = \frac{1}{n} \sum_{i=1}^{n} \Big(\text{MSE}(P, \hat{P}) - \mu\Big)^2, \tag{8}$$

where μ is the mean of the MSE over time. If a Trojan or side-channel attack modifies the power consumption pattern, the MSE and the variance σ_P^2 will increase.

The anomaly is detected if both the MSE and variance exceed their respective thresholds:

$$\text{MSE}(P', \hat{P}') > \epsilon \quad \text{and} \quad \sigma_{P'}^2 > \epsilon_v, \tag{9}$$

where P' is the modified power consumption pattern and ϵ_v is the variance threshold. The probability of an attack, P_{SC}, increases proportionally to the variance:

$$P_{SC} \propto \frac{\sigma_P^2}{\epsilon_v}, \tag{10}$$

indicating that larger deviations in variance signal a higher likelihood of a side-channel attack or hardware Trojan.

A. When is SHA-256 Sufficient, and When Does Alexis Add Value?

A byte-for-byte comparison between the bitstream hash and a *trusted* golden reference is the primary integrity mechanism. In settings where such a golden is available and stable, hash comparison alone is sufficient for configuration integrity. Alexis contributes in three practical cases: (i) a golden is unavailable but multiple clean deployments yield a consensus baseline; (ii) benign toolflow variability (e.g., compression, PR region churn) changes hashes while preserving regular structure that Alexis learns; and (iii) operational anomalies correlated with configuration irregularities. We therefore bound the overall failure probability as

$$P_T \leq_{\text{SHA-256}} + P(\text{autoencoder evasion}),$$

and we treat cryptography as the first line of defense, with Alexis as a complementary detector for structure-preserving deviations and for contexts lacking a stable golden.

VI. Experiment Design

This section describes the experimental setup to evaluate the proposed approach for anomaly detection in bitstream data generated by FPGAs. We used two types of FPGAs to ensure the variation, in which a total number of eight FPGA devices were used from which four were Digilent Basys Artix-7 FPGAs with 18 KB Fast block RAM and 32 MB flash memory, while four of them were Xilinx Zynq-7000 SoC XC7Z020 FPGAs with 2.1 MB block RAM and 512 MB flash memory, bitstream data was collected from each device. The collected bitstream data was then decrypted into a hexadecimal format for further analysis.

To create a realistic testing scenario, three of the collected bitstream files were tagged as normal or anomaly-free, while we artificially inserted hardware Trojans into two devices to alter the behavior of the FPGA and generate unexpected output.

These two bitstream files were intentionally designed to contain hardware Trojans. We designed code snippets written in Verilog specifically according to the functionality of the device and injected them before the synthesis process, which resulted in overwriting the previous output flags and altering the FPGA functionality. Moreover, two bitstreams from FPGAs that were running different applications were also used to test the proposed technique.

1) *Data Acquisition:* Obtained the original bitstream data X generated by the FPGA.
2) *Initialization:*
   a) Initialized X' to an empty matrix.
   b) Set a counter c to zero.
   c) Initialized an empty list/hash table H for storing hashed values.
3) *Bitstream Analysis:* Using the entropy-ranked selector, obtain the index set $\mathcal{I}_K$. For each $i \in \mathcal{I}_K$, extract the corresponding bitstream segment and compute its hash h_i; append h_i to the container H (thus $c \leftarrow |H|$).
4) *Checksum Calculation:* Compute the checksum C over the K hashed values in H (e.g., the MSE-based aggregation in Algorithm 2).
5) *Autoencoder Application:* Applied the autoencoder $A(X)$ to the input bitstream X:

$$\text{X'} = A(X)$$

6) *Reconstruction Error Calculation:* Calculated the reconstruction error ϵ between the original bitstream X and the reconstructed bitstream X' using the MSE loss function:

$$\epsilon = L_{\text{MSE}}(X, X') = \frac{1}{N}\sum_{i=1}^{N}(X_i - X'_i)^2 \tag{11}$$

7) *Trojan Detection:* Checked if the calculated checksum C matches the expected checksum of the Golden Reference bitstream R_g. If $C \neq R_g$, it indicates the presence of a potential Trojan in the FPGA bitstream.
8) *Threshold-Based Decision:*
   a) Calculated the maximum reconstruction error M across the bitstream:

$$M = \max(\epsilon) \tag{12}$$

   b) Compared M with a predefined threshold T to determine if the FPGA bitstream contains anomalies: If $M > T$, the FPGA is tagged as compromised (anomaly detected). Otherwise, the FPGA is considered normal (no anomaly detected).
9) *Result Output:* Output the result of the Trojan detection process, including the location and nature of the Trojan if applicable.

We tested the proposed technique against intentionally modified bitstreams for the same application as well as bitstreams for other applications. Overall, this experimentation was designed to assess the effectiveness of the proposed approach in detecting anomalies in bitstream data generated by FPGAs. By using a combination of real-world and artificial bitstream data, the performance of the auto-encoder model was evaluated under different scenarios to ensure its robustness and effectiveness. The dataset used consists of FPGA bitstreams collected from both verified and Trojan-injected configurations. Normal bitstreams represent trusted operations, while anomalous ones include injected hardware Trojans to simulate tampering. This diverse dataset provides a solid basis for evaluating Alexis' detection capabilities. To ensure the reproducibility of our results, we have provided access to our dataset [1] and code [2] at Github.

VII. Results and Discussion

This approach integrates data preprocessing, model definition, training with early stopping, evaluation, and anomaly detection into a cohesive workflow. The use of an autoencoder for anomaly detection is effective in scenarios where understanding the normal operation pattern is crucial, as it learns to reconstruct the input data and can highlight deviations from the learned normal patterns through reconstruction error. The dataset is split into training and validation sets for model development and tuning. We convert the sets to PyTorch tensors and normalize features using the training-set mean and standard deviation. For attestation, we select the entropy-ranked top-K entries (indices $\mathcal{I}_K$) and compare their hashed representations to the golden reference. The Mean Squared Error (MSE) over these K entries serves as a checksum; an anomaly is flagged when the MSE exceeds a validation-chosen threshold τ. An autoencoder model is defined with both encoder and decoder components. The encoder compresses the input into a lower-dimensional representation, and the decoder reconstructs the input from this representation. The model uses ReLU (13) activation functions and dropout for regularization and is trained using the Adam optimizer (14) and MSE (15) loss function. Training involves forward passes to compute losses on both training and validation data, followed by backpropagation to adjust the model weights.

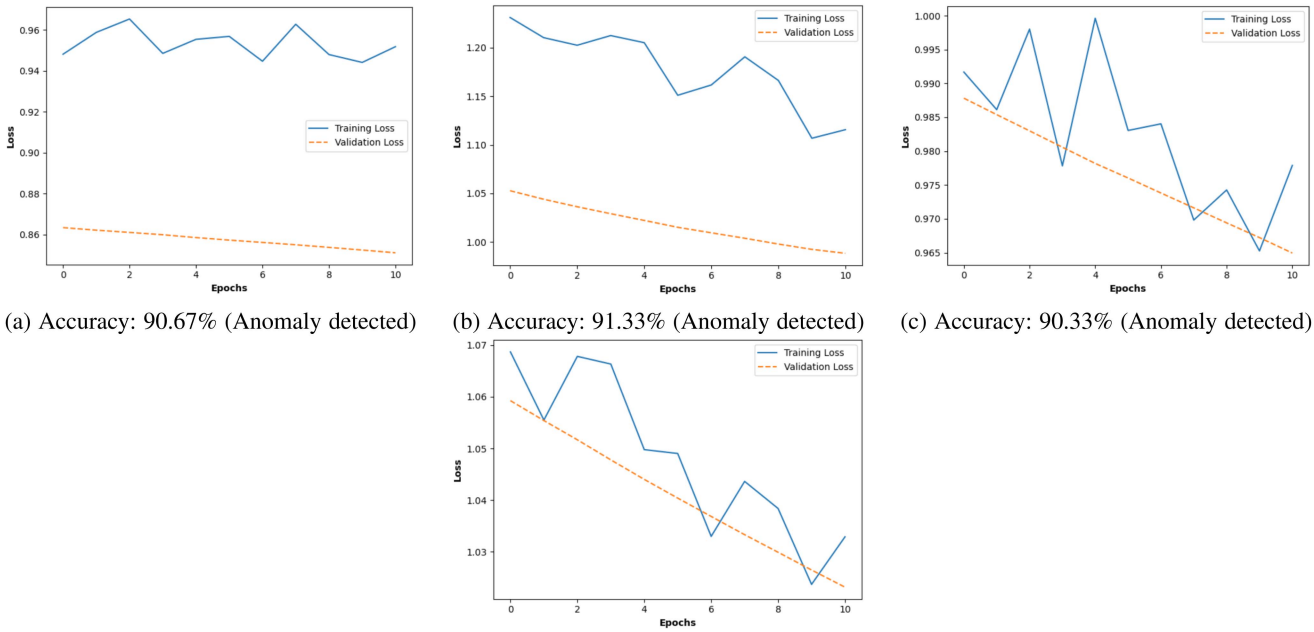
$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise,} \end{cases} \tag{13}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{14}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{15}$$

The autoencoder was trained for a specified number of epochs (100 in this case), and the loss and reconstruction errors were recorded at each iteration. Early stopping is implemented to halt training if the validation loss does not improve significantly over a set number of epochs (patience), preventing overfitting. The model's performance is evaluated on the validation data using MSE and Mean Absolute Error (MAE) (16) loss metrics. A dynamic threshold for anomaly detection is calculated based on the interquartile range (IQR) of MSE values from the validation data. Accuracy is determined by the percentage of validation

[1] https://dx.doi.org/10.21227/aqc1-dv65
[2] https://github.com/uhanif6/Alexis



(a) Accuracy: 90.67% (Anomaly detected)

(b) Accuracy: 91.33% (Anomaly detected)

(c) Accuracy: 90.33% (Anomaly detected)

(d) Accuracy: 88.33% (No Anomaly detected)

Fig. 7. Bitstream files testing results.

TABLE II
NOTATIONS

Notation	Description
X	Original bitstream data generated by an FPGA
X'	Reconstructed bitstream data obtained from the autoencoder
R_g	Golden Reference bitstream data
E	Encoder function of the autoencoder
D	Decoder function of the autoencoder
L_{MSE}	Mean Squared Error (MSE) loss function
T	Threshold value for determining anomalies
$A(X)$	Function to apply the autoencoder to input X
ϵ	Reconstruction error
M	Maximum error across the bitstream

data points whose MSE is below this dynamic threshold.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|\,. \tag{16}$$

Finally, we plotted the training and validation loss over epochs to visualize the training progress. We performed experiments on various bitstream files categorized as Normal or malicious. Fig. 7(a) to (d) show the results of the bitstream files declared malicious or normal by the auto-encoder model with accuracies.

We evaluated the performance of our proposed method using three evaluation metrics: Accuracy, MSE, MAE. Besides it, we also calculated time and memory overhead to provide analysis of Alexis computation overhead. Table III presents the performance of our method for four different sets of data, each with different characteristics. The first metric, Accuracy, measures the proportion of correctly identified anomalies or compromised devices. The table shows that the proposed method achieves

TABLE III
PERFORMANCE OF THE PROPOSED TECHNIQUE ON BITSTREAM FILES

Metric	Bitstream 1	Bitstream 2	Bitstream 3	Bitstream 4
Accuracy	90.67%	91.33%	90.33%	88.33%
MSE	0.9141	0.9316	0.9169	0.9851
MAE	0.6683	0.6804	0.7125	0.7401
Checksum- Comparison	20.7066	20.7066	20.5883	0.0000
Time Overhead (Sec)	0.0210	0.0175	0.0200	0.0200
Memory Overhead (MiB)	1.17	1.14	1.16	1.26

TABLE IV
PERFORMANCE OF SACHA ON BITSTREAM FILES

Metric	Bitstream 1	Bitstream 2	Bitstream 3	Bitstream 4
Accuracy	77.93%	86.92%	77.49%	77.82%
Altered segments	11478	6801	11707	11535

an accuracy of 91.33%, 90.67%, 90.33%, and 88.33% for the four different data sets, respectively. The second metric, MSE, measures the average of the squared differences between the predicted and actual values. while the third metric, MAE, measures the average of the absolute differences between the predicted and actual values, and fifth and sixth metrics provides the model's time and memory overheard which makes it a suitable choice to be used in resource constrained environments.

To benchmark Alexis against the state-of-the-art, we compared its performance to the most relevant and recent existing technique known as SACHa. The results obtained from SACHa when tested on the same dataset of anomalous bitstreams are summarized in Table IV.

Our results demonstrate that Alexis outperforms SACHa across multiple key metrics. Specifically, when evaluating the

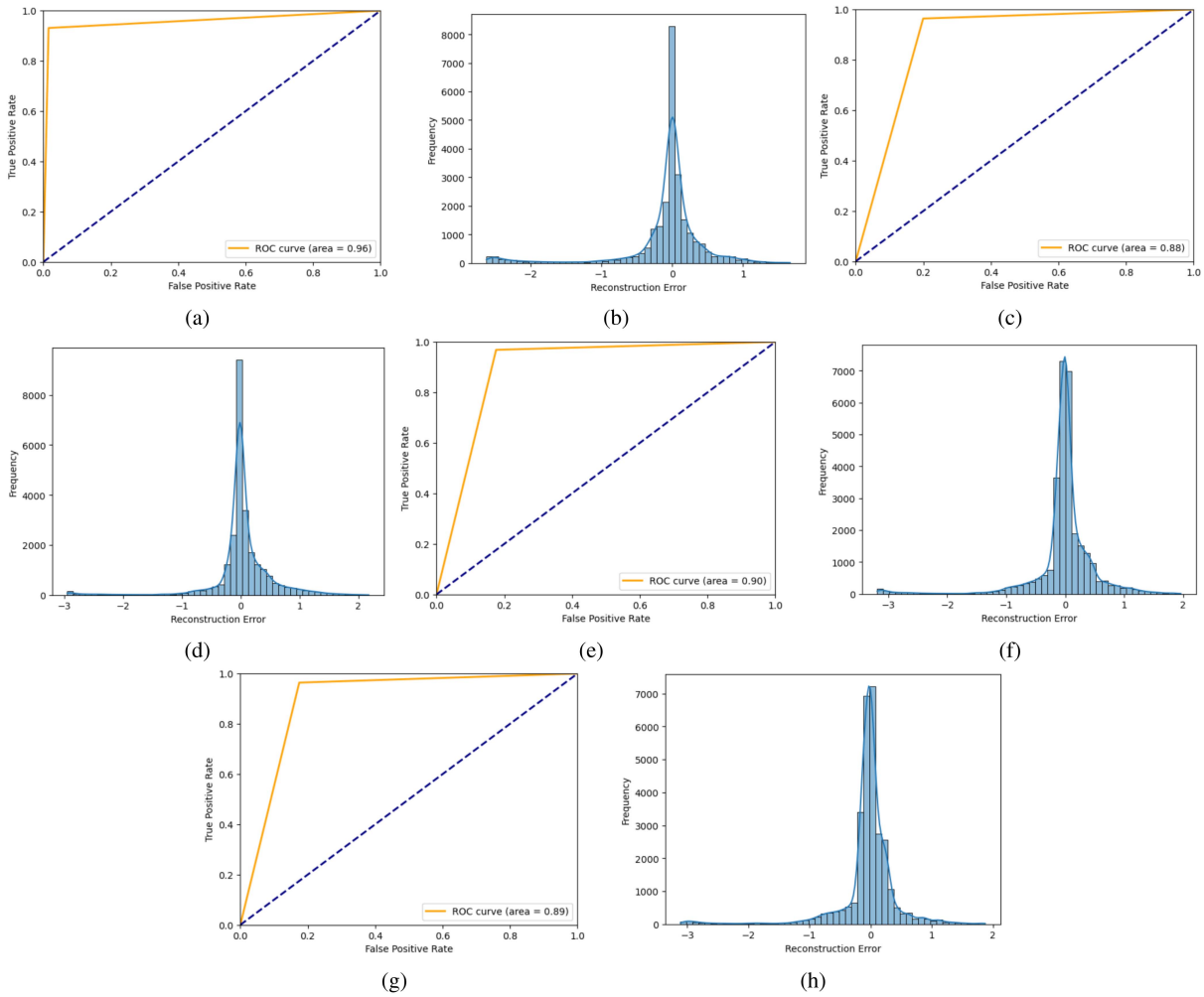


Fig. 8. Receiver operating characteristic (ROC) and reconstruction error distribution (RED).

accuracy, which measures the proportion of correctly identified anomalies or compromised devices, Alexis consistently achieves significantly higher accuracy rates compared to SACHa. For example, in Bitstream 1, our method attains an accuracy of 90.67%, whereas SACHa achieves 77.93%.

The superiority of Alexis is further illustrated by the distribution of reconstruction errors in Fig. 8(b)-(h). These figures show histograms of the reconstruction errors for four different bitstreams. The sharp peaks around zero in the histograms indicate that Alexis effectively reconstructs normal data, while the tails represent the anomalies that are detected with high sensitivity. In contrast, a broader distribution would suggest poorer anomaly detection performance, which is more characteristic of the results obtained with SACHa.

Additionally, the Receiver Operating Characteristic (ROC) curves presented in Fig. 8(a)-(g) reinforce the efficacy of Alexis. The ROC curves for each bitstream consistently show a high area under the curve (AUC), with values such as 0.96 in Bitstream 1, indicating a strong true positive rate and a low false positive rate. This performance is a direct result of Alexis's ability to capture complex relationships within the bitstream data using autoencoder models.

Furthermore, the Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics provide insights into the quality of anomaly detection. Our method consistently exhibits lower MSE and MAE values, indicating that it can more accurately assess the differences between predicted and actual values, thus achieving better precision in detecting anomalies. Alexis outperforms SACHa across these four anomalous bitstreams because the key to our method's success lies in its utilization of autoencoder models, which are adept at capturing complex relationships within the bitstream data. By training on a large dataset and effectively reducing the dimensionality of the input data, our autoencoder-based approach can discern subtle deviations in bitstream patterns, leading to more accurate and sensitive detection of anomalies. In contrast, SACHa lacks the fine-grained analysis capabilities that our method offers, potentially resulting in lower accuracy and more instances of false negatives in anomaly detection.

Overall, the experimental results, supported by both the reconstruction error distributions and the ROC curves, clearly demonstrate the superiority of Alexis in detecting anomalies and compromised FPGA devices using bitstream data.

TABLE V
BEST FACTORS OF ALEXIS IN COMPARISON WITH EXISTING TECHNIQUES

Factors	SWATT [3]	Fuzzing HW like SW [10]	SACHa [2]	Alexis
Anomaly Detection Technique	Software-based	Fuzzing-based	Hardware-based	Autoencoder-based
Use of Golden Reference	Yes	Yes	Yes	Yes
Hardware Trojan Detection	Yes	Yes	Yes	Yes
Resilience against Side-Channel Attacks	Limited	No	Limited	**Partial**
Utilization of Machine Learning	No	No	No	**Yes**
Use of PUFs	No	No	**Yes**	**No**
Integration of Auto-encoders	No	No	No	Yes
Configurability	Medium	Medium	Low	**High**
Performance Evaluation	Limited	Limited	Limited	**Comprehensive**
Practical Applicability	Yes	Yes	Limited	Yes
Flexibility for FPGA Types	Yes	Limited	Yes	Yes
Robustness against Trojan Evasion	Limited	No	Limited	**Comprehensive**

A. Feature-Based Comparison

Table V provides a detailed comparison of Alexis with SWATT [3], SACHa [2], and Fuzzing [10], showcasing the distinctive advantages and capabilities that set Alexis apart in the field of FPGA security and anomaly detection.

Alexis employs an innovative autoencoder-based anomaly detection technique, allowing accurate identification of anomalies in bitstream data, a distinct approach compared to SWATT's software-based and SACHa's hardware-based methods. While Alexis offers partial resilience, SWATT and SACHa have limited resilience against side-channel attacks. The fuzzing approach doesn't address this issue. Our approach stands out for effectively utilizing machine learning. Alexis' integration of autoencoders provides advanced feature extraction, improving detection accuracy. Its flexibility in handling different FPGA types is notable, unlike SWATT and SACHa's limited flexibility while Fuzzing shares this trait. Alexis, albeit comprehensively, demonstrates a notable level of resilience against attempts at Trojan evasion, highlighting its substantial defense capabilities in this regard. SWATT and SACHa demonstrate similar limitations, while Fuzzing lacks this feature.

VIII. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we presented *Alexis*, a novel methodology for anomaly detection in FPGA-generated bitstream data using an autoencoder-based approach. By employing an encoder-decoder model and comparing specific bitstream locations to a golden reference, *Alexis* effectively verifies FPGA integrity. Our experimental results showed that *Alexis* accurately identifies compromised FPGAs, demonstrating high accuracy, minimal MSE loss, and low MAE error, thereby showcasing its effectiveness in enhancing FPGA security. Nonetheless, the dependency on an easily accessible golden reference (*Rg*) presents a limitation in practical scenarios. To address this, future work could involve adapting *Alexis* for detecting hardware Trojans in additional components, such as microprocessors and memory, broadening its scope. Moreover, integrating machine learning techniques to automate the detection and classification of Trojans offers a promising pathway to develop a more robust security system for cyber-physical systems.

A. Deployment, Scope & Adaptive Strategies

Alexis is deployed per device family/SKU using an entropy-ranked top-K feature template and a family-level autoencoder trained on multiple clean configurations. When a trusted golden bitstream is available, byte-for-byte hash comparison remains the *primary* integrity check; Alexis acts as a complementary detector for structure-preserving drift and operational irregularities. In golden-limited settings, operators or OEM/integrators can build a consensus baseline from multiple clean units of the same SKU/family and maintain a small catalog of family/SKU models (e.g., Artix-7/Basys3 and Zynq-7000), rather than per-design models. Because Alexis analyzes compiled configurations, identifier-only HDL edits that do not alter the synthesized netlist are not flagged, whereas functional edits that modify configuration frames are. Limitations include inability to detect runtime-only Trojans when the configured bitstream matches a trusted golden and potential susceptibility to adaptive evasion. Practical mitigations include pairing Alexis with runtime attestation and hardware roots of trust, periodic re-training with fresh clean data, and lightweight ensembles/threshold audits. Looking ahead, Alexis can be integrated into existing IDS workflows as an integrity layer and enhanced with AI-driven adaptive strategies that leverage real-time telemetry (e.g., network traffic and resource utilization) to anticipate emerging threats and mitigate DoS-style conditions through anomaly-aware monitoring and response.

ACKNOWLEDGMENT

We express our sincere gratitude to Mr. Jo Vliegen from KU Leuven for generously sharing the SACHa codebase with us and permitting us to adapt it to our dataset.

REFERENCES

[1] M. N. Aman et al., "HAtt: Hybrid remote attestation for the Internet of Things with high availability," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7220–7233, Aug., 2020.

[2] J. Vliegen, M. M. Rabbani, M. Conti, and N. Mentens, "A novel FPGA architecture and protocol for the self-attestation of configurable hardware," *IACR Cryptol. ePrint Arch.*, 2019, Art. no. 405. [Online]. Available: https://eprint.iacr.org/2019/405

[3] A. Seshadri, A. Perrig, L. Van Doorn, and P. Khosla, "SWATT: Software-based attestation for embedded devices," in *Proc. IEEE Symp. Secur. Privacy*, 2004, pp. 272–282.
[4] 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 Things J.*, vol. 9, no. 20, pp. 20431–20443, Oct., 2022.
[5] M.M. Rabbani, J. Vliegen, J. Winderickx, M. Conti, and N. Mentens, "SHeLA: Scalable heterogeneous layered attestation," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10240–10250, Dec., 2019.
[6] C. Basile, S. Di Carlo, and A. Scionti, "FPGA-based remote-code integrity verification of programs in distributed embedded systems," *IEEE Trans. Syst., Man, Cybern., Part C (Appl. Rev.)*, vol. 42, no. 2, pp. 187–200, Mar., 2012.
[7] A. Viticchié, C. Basile, A. Avancini, M. Ceccato, B. Abrath, and B. Coppens, "Reactive Attestation: Automatic detection and reaction to software tampering attacks," in *Proc. ACM Workshop Softw. Protection*, Oct., 2016, pp. 73–84.
[8] Ü. Koçabas, A.R. Sadeghi, C. Wachsmann, and S. Schulz, "Poster: Practical embedded remote attestation using physically unclonable functions," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, Oct. 2011, pp. 797–800.
[9] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, no. 1, 2023, Art. no. 1.
[10] T. Trippel, K. G. Shin, A. Chernyakhovsky, G. Kelly, D. Rizzo, and M. Hicks, "Fuzzing hardware like software," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 3237–3254.
[11] S. K. Saha and C. Bobda, "FPGA accelerated embedded system security through hardware isolation," in *Proc. Asian Hardware Oriented Secur. Trust Symp.*, 2020, pp. 1–6.
[12] A. Chakraborty and A. Srivastava, "Hardware-software co-design based obfuscation of hardware accelerators," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2019, pp. 547–552.
[13] N. Karimi, K. Basu, C.-H. Chang, and J. M. Fung, "Hardware security in emerging technologies: Vulnerabilities, attacks, and solutions," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 2, pp. 223–227, Jun., 2021.
[14] E. Singh et al., "A-QED verification of hardware accelerators," in *Proc. 57th ACM/IEEE Des. Automat. Conf.*, 2020, pp. 1–6.
[15] L. Deng, C. Wu, D. Lian, and M. Zhou, "Transposed variational auto-encoder with intrinsic feature learning for traffic forecasting," 2022, *arXiv:2211.00641*.
[16] X. Chen, L. Wang, Y. Wang, Y. Liu, and H. Yang, "A general framework for hardware Trojan detection in digital circuits by statistical learning algorithms," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1633–1646, Oct., 2017.
[17] L.E. Olson, S. Sethumadhavan, and M.D. Hill, "Security implications of third-party accelerators," *IEEE Comput. Archit. Lett.*, vol. 15, no. 1, pp. 50–53, Jan.–Jun., 1 2016.
[18] J. Zhang and G. Qu, "Recent attacks and defenses on FPGA-based systems," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 12, no. 3, pp. 1–24, 2019.
[19] M. Rathor and A. Sengupta, "Obfuscating DSP hardware accelerators in CE systems using pseudo operations mixing," in *Proc. Zooming Innov. Consum. Technol. Conf.*, May 2020, pp. 218–221.
[20] A. Sengupta, M. Rathor, S. Patil, and N.G. Harishchandra, "Securing hardware accelerators using multi-key based structural obfuscation," *IEEE Lett. Comput. Soc.*, vol. 3, no. 1, pp. 21–24, 1 Jan./Jun., 2020.
[21] A. Singla, A. Mudgerikar, I. Papapanagiotou, and A.A. Yavuz, "HAA: Hardware-accelerated authentication for Internet of Things in mission critical vehicular networks," in *Proc. IEEE Mil. Commun. Conf.*, Oct. 2015, pp. 1298–1304.
[22] B.Y. Huang, S. Ray, A. Gupta, J. M. Fung, and S. Malik, "Formal security verification of concurrent firmware in SoCs using instruction-level abstraction for hardware," in *Proc. 55th Annu. Des. Automat. Conf.*, Jun. 2018, pp. 1–6.
[23] M. I. M. Collantes and S. Garg, "Do not trust, verify: A verifiable hardware accelerator for matrix multiplication," *IEEE Embedded Syst. Lett.*, vol. 12, no. 3, pp. 70–73, Sep., 2020.
[24] S. Hristozov, M. Huber, and G. Sigl, "Protecting RESTful IoT devices from battery exhaustion DoS attacks," in *Proc. IEEE Int. Symp. Hardware Oriented Secur. Trust*, Dec. 2020, pp. 316–327.
[25] A. Lasheras, R. Canal, E. Rodrıguez, and L. Cassano, "Securing RSA hardware accelerators through residue checking," *Microelectronics Rel.*, vol. 116, 2021, Art. no. 114021.
[26] S. Chattopadhyay et al., "Scaling up hardware accelerator verification using A-QED with functional decomposition," in *Proc. Formal Methods Comput. Aided Des.*, 2021, pp. 42–52.
[27] C. Pilato, S. Garg, K. Wu, R. Karri, and F. Regazzoni, "Securing hardware accelerators: A new challenge for high-level synthesis," *IEEE Embedded Syst. Lett.*, vol. 10, no. 3, pp. 77–80, Sep., 2018.
[28] M. N. Aman, M. H. Basheer, and B. Sikdar, "Data provenance for IoT with light weight authentication and privacy preservation," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10441–10457, Dec., 2019.
[29] Y. Liu, X. Tao, X. Li, A. Colombo, and S. Hu, "Artificial intelligence in smart logistics cyber-physical systems: State-of-the-arts and potential applications," *IEEE Trans. Ind. Cyber- Phys. Syst.*, vol. 1, pp. 1–20, 2023.
[30] Y. Jiang, S. Wu, R. Ma, M. Liu, H. Luo, and O. Kaynak, "Monitoring and defense of industrial cyber-physical systems under typical attacks: From a systems and control perspective," *IEEE Trans. Ind. Cyber- Phys. Syst.*, vol. 1, pp. 192–207, 2023.
[31] Y. Dai, M. Li, K. Zhang, and Y. Shi, "Robust and resilient distributed MPC for cyber-physical systems against DoS attacks," *IEEE Trans. Ind. Cyber-Phys. Syst.*, vol. 1, pp. 44–55, 2023.
[32] I. K. M. Jais, A. R. Ismail, and S.Q. Nisa, "Adam optimization algorithm for wide and deep neural network," *Knowl. Eng. Data Sci.*, vol. 2, no. 1, pp. 41–46 2019.
[33] S. Yoganand, S. S. Babu, and C. Madhu, "FPGA based RADAR signal emulator for signal processing test applications," *Int. J. Eng. Res. Appl.*, vol. 4, pp. 362–367, 2014.
[34] S. Aslam, "Design for prognostics and security in field programmable gate arrays (FPGAs)," Cranfield Univ., Bedford, U.K., 2020.
[35] COREnect Consortium, "European core technologies for future connectivity systems and components," *CORDIS EU Res. Results, Horiz. 2020 Project Rep.*, Eur. Commission, 2022. [Online]. Available: https://cordis.europa.eu/project/id/956830/results
[36] "FPGA in aerospace and defense: Advancements and applications," 2023. [Online]. Available: https://fpgainsights.com/fpga/fpga-in-aerospace-and-defense-advancements-and-applications/
[37] AMD, "CSIRO case study," 2025. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/resources/case-studies/csiro-case-study.pdf
[38] AMD, "CERN accelerates innovation with xilinx FPGAs. PDF. AMD," 2025. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/resources/case-studies/cern-case-study.pdf
[39] Y. Qiu et al., "Variational autoencoder-assisted unsupervised hardware fingerprint authentication in a fiber network," *Opt. Lett.*, vol. 49, no. 8, pp. 2029–2032, 2024.
[40] E. Mattei et al., "Feature learning for enhanced security in the Internet of Things," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2019, pp. 1–5.
[41] M. Zamini and G. Montazer, "Credit card fraud detection using autoencoder based clustering," in *Proc. 9th Int. Symp. Telecommun.*, Dec. 2018, pp. 486–491.
[42] Z. Cheng, S. Wang, P. Zhang, S. Wang, X. Liu, and E. Zhu, "Improved autoencoder for unsupervised anomaly detection," *Int. J. Intell. Syst.*, vol. 36, no. 12, pp. 7103–7125 2021.

Umara Hanif received the B.Sc. degree in computer science from the University of Engineering & Technology, Lahore, Pakistan, in 2020. She is currently working toward the Ph.D. degree in electrical and computer engineering with the National University of Singapore, Singapore. She has industry experience as a Cybersecurity Engineer with VaporVM, Lahore, focusing on fortifying digital infrastructures, conducting vulnerability assessments, and implementing proactive security strategies. She was a Cybersecurity Research Analyst with Vultara, Inc., Detroit, focusing on IoT automotive and device security. Her research interests include hardware security, IoT device security, and the application of machine learning in anomaly detection and cybersecurity.

Muhammad Naveed Aman (Senior Member, IEEE) received the B.Sc. degree in computer systems engineering from KPK UET, Peshawar, Pakistan, the M.Sc. degree in computer engineering from the Center for Advanced Studies in Engineering, Islamabad, Pakistan, in 2006, and the M.Engg. degree in industrial and management engineering and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2008, and 2012, respectively. He is currently an Assistant Professor with the University of Nebraska-Lincoln, Lincoln, NE, USA. His research interests include IoT and network security, hardware systems security and privacy, wireless and mobile networks, and stochastic modeling.

Biplab Sikdar (Fellow, IEEE) received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He was an Assistant Professor from 2001 to 2007 and Associate Professor from 2007 to 2013 with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA. He is currently a Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, where he is also the Head of the Department of Electrical and Computer Engineering and Director of Cisco-NUS Corporate Research Laboratory. His research interests include IoT and cyber-physical system security, network security, and network performance evaluation. He was the recipient of NSF CAREER award, Tan Chin Tuan fellowship from NTU Singapore, Japan Society for Promotion of Science fellowship, and Leiv Eiriksson fellowship from the Research Council of Norway. Dr. Sikdar is a member of Eta Kappa Nu and Tau Beta Pi. He was an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE INTERNET OF THINGS JOURNAL, IEEE COMSOC, VTS Distinguished Lecturer, and ACM Distinguished Speaker.