International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 10-No.4 (2024) pp. 1372-1378
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

ProTECT: A Programmable Threat Evaluation and Control Unit for Zero Trust Networks

Rahul SHANDILYA[1]*, R.K. SHARMA[2]

[1]School of VLSI Design and Embedded Systems, NIT Kurukshetra-136119, Haryana, India
* **Corresponding Author Email:** rss.nitk@gmail.com - **ORCID:** 0009-0006-3073-092X

[2]Dept. of Electronics and Communication Engineering, NIT Kurukshetra-136119, Haryana, India
Email: mail2drks@gmail.com - **ORCID:** 0009-0009-2430-669X

Article Info:

DOI: 10.22399/ijcesen.673
Received : 21 November 2024
Accepted : 27 November 2024

Keywords :

Device Security,
Threat Monitoring,
Network Security,
Zero Trust Architectures,
FPGA.

Abstract:

As Zero Trust Architectures (ZTA) become increasingly adopted in enterprise networks, so it is essential to continuously monitor the security status of connected devices. Real-time threat monitoring within the devices that are connected to the network is necessary for this. It's challenging, especially in resource-constrained settings, to ensure continuous monitoring in current devices available in market. ProTECT (**Pro**grammable **T**hreat **E**valuation and **C**on**t**rol Unit) addresses this challenge by providing a continuous real-time monitoring, non-tamperable, trust score for ZTA network connected devices. Trust score in security coprocessor segregated from device computing architecture has been determined employing real-time hardware monitoring of CPU micro-architectural signals. We examined ProTECT on an open-source RISC-V processor based architecture against ransomware, RoP & cache-based micro-architectural attacks. While illustrating area overheads, we implement framework on an AMD Virtex XC7V2000T FPGA Module.

1. Introduction

With the advent of IoT, contemporary network infrastructure is becoming dynamic, accommodating growing array of distributed devices, including wearable technology, including health monitors & smartwatches, as well as sensor devices in essential infrastructure. Internet-connected devices proliferation has significantly escalated cyber-attacks threat. Weak, discretionary, perimeter-focused access policies employed in contemporary networks exacerbate these threats. NIST established specifications [1-6] for ZTA for reducing these threats, wherein device's network access privileges are determined dynamically by its behavior rather than being statically assigned depending on user authentication. Consequently, device exhibiting anomalous behavior will have its network privileges dynamically restricted.

Fundamental requisite for reliable implementation of ZTA is continuous monitoring of device's security status during application execution. Network must determine if device is compromised or acting as specified. Runtime health signature acquisition is difficult. Native methods could remotely monitor device network activity or install software daemon for recording operations for anomalies. The previous method failed for yielding adequate insights into detailed activities. Device may fail to identify all malicious behaviors, & in latter scenario, trusting received signatures becomes challenging, particularly when device is compromised.

Current paper proposes runtime security-health implementation monitoring for devices through specialized hardware modules that monitor System-on-Chip (SoC) operation for identifying attack-specific behaviors. Monitoring is conducted at hardware level software stack irrespectively. Hardware digitally verifies device trust score & transmits it to Zero Trust Network (ZTN) host for confidentiality along with integrity. This enables network for dynamically adjusting device access policies. Contrary to network as well as software-based methods, device hardware calculates trust scores, providing accurate and reliable security indicators.

Illustrated in figure 1, our framework, termed ProTECT (**Pro**grammable **T**hreat **E**valuation and **C**on**t**rol Unit), comprises specialized hardware modules integrated within SoC. Event Monitor

modules are intricately connected to CPU cores bus interfaces with monitor configurable execution parameters. A secure RISC-V processor is responsible for executing the architecture-specific Threat Evaluation Model (TEM) firmware that is stored in flash ROM memory and provide the trust score value for the session. Trust scores that measure device's security health are calculated by TEM after it periodically retrieves execution logs from trace memory.

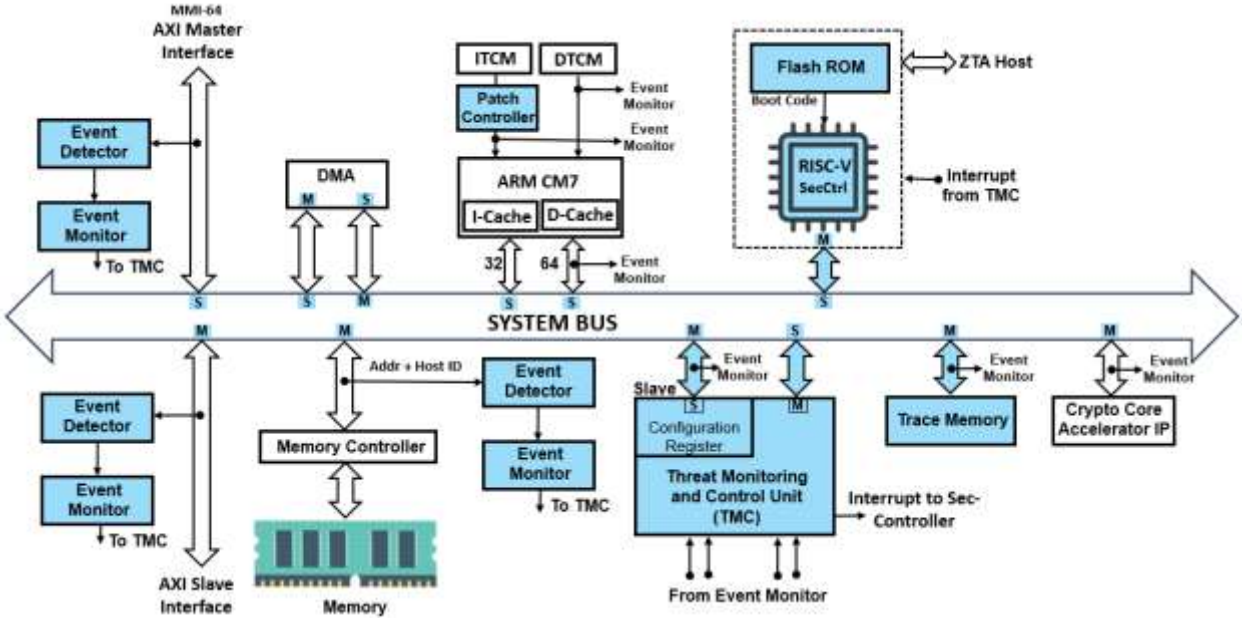

Figure 1. *ProTECT framework*

Applications executing on CPU cores are not allowed to accessing secure elements used by sec-controller for tracking critical signals of each CPU core, computing trust scores, evaluating trace memory, for communicating via network interface. This assures trust score's integrity and reliability. Following are paper's primary contributions:

• ProTECT framework can be integrated into SoCs thereby providing devices in zero-trust networks with non-tamperable, confidential, & trustworthy trust scores, enabling them for determining dynamic access policy decisions.

• ProTECT is demonstrated by using TEMs that utilize ANNs (Artificial Neural Networks) & SVM(Support Vector Machines) to RISC-V-based SoC [10-14]. For variety of workloads, including embench-IoT suite [15], Coremark [16], & Dhrystone [13], as well as threat vectors involving micro-architectural side-channels, RoP(Return-oriented Programming) attacks, & ransomware attacks, TEMs calculate trust scores.

• The area overheads of ProTECT were determined based on the synthesis analysis of AMD Virtex XC7V2000T FPGA.

The remainder of the paper is arranged in the following way: Section 2 describes related work, and Section 3 explains ProTECT architecture. Section 4 contains the ProTECT assessment and Section 5 contains the conclusion.

2. Related Works

ZTA network security monitoring solutions must fulfill these criteria: (a) monitors configurability, (b) adaptability to emerging threat vectors, (c) monitoring reliability (d) isolation from attack surface. Device-level behavioral monitoring solutions [5, 11, 7, 10] employ HPCs (Hardware Performance Counters) [8] prevalent in contemporary microprocessors. Micro-architectural events are logged with constrained configurability [8, 9]. Additionally, HPC-based monitoring's isolation & dependability are limited by its reliance on operating system(OS) for event reading. Further research [5, 11, 7] attempted for minimizing HPCs limitations through developing monitoring mechanisms with focus on security. Kuruvila et al. [5] reported developing appropriate customized counters through instruction sequences application. Monitoring & threat assessment operate as applications without isolating SoC, a primary drawback of these solutions. Recent investigations [11, 7], including PHMon, enhance isolation through the utilization of specialized hardware for monitoring. However, configuring mechanism for monitoring necessitates OS support, hence add the overhead for host. Moreover, PHMon [7] exclusively monitors executed instructions, thereby lacking its ability to address intricate threat vectors,

including ransomware or side-channel attacks. The current existing design lacks the feature that calculates trust levels for each process at runtime and does not take into account threats that were detected previously. These solutions are contrasted with ProTECT in table 1. Utilizing a dedicated hardware monitor along with separate dedicated security co-processor other than primary core in SoC, ProTECT improve the isolation limitations, monitoring, adaptability, along with configurability.

3. The ProTECT Framework

Illustrated in figure 1, ProTECT framework consists of three essential components. 1) The Threat Monitoring and Control Unit (TMC) conducts runtime surveillance for every designated CPU-core interface and generates behavioral traces. TMC stores intermediate data and traces on Trace memory. The events detector and monitor are set up in accordance with a pre-determined configuration. 2) Trace memory is designated memory unit for every monitor for recording observed events. Only TMC and Sec-Controller are granted reading and writing privileges to trace memory, ensuring its isolation from remaining memory units within SoC. 3) Sec-Controller is a security coprocessor capable of reading each trace memory's logs then invoking the TEMs on them. Sec-Controller offers device trust score based on output from TCMs & trace memory's logs. We discuss in great detail every such element in this section.

3.1 Event Monitor and Trace Memory

Program execution generally follows a phase behavior [12], where each phase is associated with specific micro-architectural component events. As a result, every attack leaves behind unique fingerprint in hardware state that utilized for recognizing attacks & gauge their threat. To identify these fingerprints, ProTECT's event monitor offers fine-grained runtime monitoring. Events in event monitoring are selected according to threat model, contrary to HPCs, that have not been developed for security applications therefore aren't customizable [8,9]. For instance, we define threat model in our ProTECT implementation, including threat vectors, RoP attacks, ransomware, as well as micro-architectural side-channels. We established events to track memory access instructions patterns, along with cache parameters, on basis of this threat model with only minor modifications of current design. Other CPU cores in SoC have no ability to impact these behavioral logs considering each core has its own event monitor.

Observed events are recorded in trace memory to establish runtime behavioral trace for running program. The frame format for trace log is shown in figure 3, each trace log incorporated with 32-bit timestamp value, trace header, host signature and trust factor. Only corresponding event monitor has write access to trace memory. The Sec-Controller master interface is granted read and write privileges, but not other masters, to ensure isolation.

3.1. Sec-Controller and Trace Memory

ProTECT framework's security coprocessor is designated as Sec-Controller. It offers an environment for TEMs in operating for determining attack-like behavior captured in trace memory and it periodically reads monitored logs of processes of each CPU core.

For adapting to an evolving threat landscape, these TEMs are trained offline then deployed on Sec-Controller in line with threat model. TEM outputs for each CPU core determine device trust score. It is metric ranging from 0 (no trust) to 10 (complete trust) quantifies security status of device within ZTA network. The ProTECT trust score's computation frequency is regulated by the epoch parameter. Our experiments indicate that threat estimation using 200 clock cycles is optimal. However, epoch value depends on the frequency of threat events. It can be configured within the TCM module to trigger the Sec-controller when exhausted. Sec-Controller operates in isolation of SoC computing stack, as demonstrated in figure 1. Sec-Controller memory has exclusive read/write access. It retains TEM code & intermediate data for trace memory unit in event processing. Inhibiting other CPU-cores for interfering in threat assessment logic. Figure 2 is the memory trace frame format.

***Figure 2**. Memory Trace frame format*

3.3. Trust Score

A device's trust score is calculated through trust factor by TEM. The trust score mechanism provides an overview of the level of trust during a ongoing session. The correlation between the behavior observed in the ongoing session and the history of the session plays a significant role in determining the level of risk involved in the session. Program behavior in each CPU core within SoC at t-th epoch determines device trust score value from 0 to 10, with lower scores indicating better security. Let the TEM output be TS_t at t-th epoch which determines the duration of current session. The trust score calculation for the current session takes into account all trust factors applied according to threat types. The result of this evaluation is a consolidated value of trust level, which describes the overall security

posture of the device. In the trust scoring calculation, trust factor are not weighed equally for all modules; the most restrictive trust factor having higher value takes precedence over all other trust factor given in table 1. The order of trust factor restrictiveness (from least restrictive to most restrictive) is as follows.

Table 1. *Trust factor value assignment based on trust level*

S.No.	Trust Level	Description	Trust Factor
1	High Trust Level	All allowed modules, no restriction	0
2	Medium Trust Level	For internal modules inside the SoC, which are not intended to access other modules, the device's Trust level will be set to medium.	1
3	Low Trust Level	Some restriction based on access types	2
4	Zero Trust Level	All access restricted	3

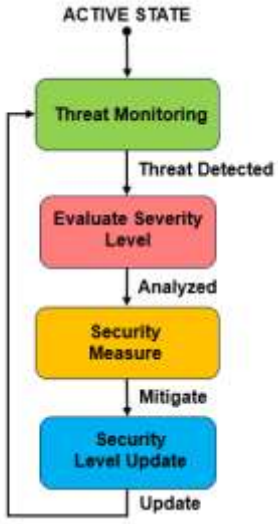

Figure 3. *Threat Evaluation Model*

The trust factor that is the most restrictive has more impact on the output of the trust scoring calculation. Protected resources cannot be accessed by the device if the trust level drops to zero. The higher trust score calculated by TEM has contributed to blocking malicious devices by design.

4. Results and Discussions

We analyzed ProTECT framework on open-source 5-stage RISC-V processor [14] SoC. Our focus is on workloads involving threat vectors, including RoP attacks, cache-based side-channel attacks, and ransomware. We configured 10 events detector event monitor to log. These events encompasses read & write requests from every core, cache line flushes, cache hits in addition to misses, with counter for RoP devices that correlates to ret & call instructions, selected based on threat vectors. On basis of their efficacy in identifying threats subset of events to monitor has been selected. We demonstrate Sec-Controller in our implementation by implementing an additional RISC-V CPU core that performs TEMs employing ANN along with customized implementations of a one-class SVM algorithm. Sec-Controller has single 36k BRAM configuration. In order to reveal area specifications, further tested ProTECT framework employing an AMD Virtex XC7V2000T FPGA Module.

4.1 Evaluating ProTECT with various threat vectors

Given consecutive attack exhibits unique execution characteristics, it is imperative to meticulously select subset of events that most effectively encapsulate threat model. Employing embench-IoT benchmark suite [15], Coremark [16], Dhrystone [13], along with Decision Forest classifier, Figure 1 demonstrates that selected events distinguish attacks from benign programs. Occurrences of read requests & cache misses are effective in categorizing ransomware attacks, whereas gadget counter event primarily identifies RoP attacks. Effectiveness of classifying cache side-channel attacks varies among different cache events. TEM training offline utilizes event traces & is configured through secure ZTA host communication channel (Figure 1). Threat assessment & trust scores for various attacks in threat model would be discussed subsequently. Ransomware, which is designed to compromise user data within the system by acquiring privileges and systematically encrypting file systems. Our approach to creating RISC-V ransomware attacks involves utilizing different cryptographic algorithms for analysis. TEM employs an offline-trained ANN with essential monitors for these attacks. This ANN comprises an input layer containing 16-nodes along with 2-hidden layers with 16 & 8 nodes, correspondingly. Microarchitectural Side-Channel Attacks can be employed including Prime+Probe & Flush+Reload [4] for extracting sensitive information, from cache memory includes passwords & cryptographic keys. We utilize cache flushing & timing measurements on RISC-V for executing these attack behaviors in our analysis [14]. Sec-Controller categories attacks One-class SVM in TEM. Current model has been trained on benign programs dataset against cache-based attacks [2]. Model clusters data on this benign behavior. A comparison of ProTECT with existing security monitoring solutions. Table 2 is comparison of ProTECT with existing security monitoring solutions.

Table 2. *A comparison of ProTECT with existing security monitoring solutions*.

Research Articles →	**Kuruvila *et al.* [5]**	**Yoon *et al.* [11]**	**Delshadtehrani *et al.* [7]**	**Nikhilesh, et al. [2]**	**ProTECT (Proposed Work)**
Event Monitor Implementation	SW	HW	HW	HW	HW
Run-time Configurable Modification Support	No	No	No	No	Yes, Run-time configurable
Monitor Programmability	Fixed Configuration	Fixed Configuration	Fixed Configuration	Fixed Configuration	Programmable
Threat Vector Adaptability	Yes	No	No	Yes	Yes
Isolation of HW Monitor	No	No	Yes	Yes	Yes
Host Off-loading	No	No	No	No	Yes, Dedicated Resources Available
Instruction Decode and Patch Controller	No	No	No	No	Yes
Continuous Event Detection Count	NA	NA	NA	Up to 15 events	Up to 512 Events capturing possible Continuously without reset (RAM Size:4K)
Real-time timestamp Capturing for events	No	No	No	No	Yes
Core Instruction Execution Monitoring	No	No	No	No	Yes

This model evaluates new program by calculating distance between event vectors relating to program as well as benign clusters. Event vectors in benign clusters indicate benign program behavior. Malicious software event vectors usually appear outside benign clusters.

ROP attacks are used to create addresses that refer back to specific points in program code for daisy-chain malicious logic. To efficiently identify these attacks, we designed an event monitor ret & call instructions to capture gadget execution.

Every device launches in trusted manner (TS_t=0). We identified benign program execution consistently maintains high trust. However, ProTECT's threat evaluation demonstrates that a device executing micro-architectural attack , ransomware, or RoP attacks consistently decreases trust which is identified with increased value of TS_t. Micro-architectural attacks, including Flush+Reload along with Prime+Probe [4], conduct anomalous cache accesses, due to this device blocks the access and move to zero trust state with increased value of TS_t to 10.

4.2. ProTECT Overheads

ProTECT doesn't impact SoC application performance since it doesn't interfere with critical path. However, incorporating hardware requires more silicon. Area overheads for ProTECT-attributed synthesized on an AMD Virtex XC7V2000T FPGA device. Table 3 presents the analysis of FPGA resource overhead that is caused by the addition of ProTECT module to the target FPGA. The ProTECT framework requires an extra 2.35% of CLBs (Configurable Logic Blocks) and 8.5% of FFs (Flip Flops) to be added in addition to the FPGA target. The proposed FIS does not have any limitations as compared to existing methods and requires less number of CLBs and FFs resources inside FPGA. It includes trace memory units for each CPU-core & event monitors. Network Security is an important subject and a number of papers have been reported on this subject [17-26].

5. Conclusions

We presented the creation, implementation, and testing of ProTECT, a framework designed to

Table 3. *FPGA resources overhead due to ProTECT module integration in sub-system design.*

Slice Logic	Sub-System without ProTECT	Sub-System with ProTECT	% Overhead
Slice LUTs	267552	273709	2.35 %
· LUT as Logic	264400	270481	2.3 %
· LUT as Memory	3152	3228	2.4 %
Slice Registers	148612	161244	8.5 %

meticulously monitor the security status of devices in the ZTA network. ProTECT offers configurable, adaptable, isolated, & tamper-resistant mechanism for fulfilling ZTA network requirements. ProTECT framework is capable of enforcing a variety of security policies at runtime and also assisting with detecting software bugs and security vulnerabilities. In the ZTA domain, ProTECT offers a novel solution that can update security policies in run-time for embedded device security, including integration of device integrity metrics into trust scores and security monitoring within the framework. Our ProTECT prototype includes a full FPGA implementation that interfaces the monitor with a RISC-V processor, along with the necessary hardware and firmware support. On average, The proposed ProTECT framework requires an extra 2.35% of CLBs (Configurable Logic Blocks) and 8.5% of FFs (Flip Flops) to be added in addition to the FPGA target, which is less as compared to existing design.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Tsai, M., Lee, S., & Shieh, S. W. (2024). Strategy for implementing of zero trust architecture. *IEEE Transactions on Reliability*. 73(1);93-100, doi: 10.1109/TR.2023.3345665

[2] Singh, N., Pal, S., Leupers, R., Merchant, F., & Rebeiro, C. (2024). PROMISE: A Programmable Hardware Monitor for Secure Execution in Zero Trust Networks. *IEEE Embedded Systems Letters*. Pp(99) Doi: 10.1109/LES.2024.3354831

[3] Federici, F., Martintoni, D., & Senni, V. (2023). A zero-trust architecture for remote access in industrial IoT infrastructures. *Electronics*, 12(3);566.

[4] Singh, N., Ganesan, V., & Rebeiro, C. (2022). Secure Processor Architectures. In Handbook of Computer Architecture (pp. 1-29). Singapore: *Springer Nature Singapore*.

[5] Kuruvila, A. P., Mahapatra, A., Karri, R., & Basu, K. (2021). Hardware performance counters: Ready-made vs tailor-made. *ACM Transactions on Embedded Computing Systems (TECS)*, 20(5s), 1-26.

[6] Stafford, V. (2020). Zero trust architecture. *NIST special publication*, *800*, *207*.

[7] Delshadtehrani, L., Canakci, S., Zhou, B., Eldridge, S., Joshi, A., & Egele, M. (2020). {PHMon}: A programmable hardware monitor and its security use cases. *In 29th USENIX Security Symposium (USENIX Security 20)* (pp. 807-824).

[8] Das, S., Werner, J., Antonakakis, M., Polychronakis, M., & Monrose, F. (2019, May). Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. *In 2019 IEEE Symposium on Security and Privacy (SP)* (pp. 20-38). IEEE.

[9] Zhou, B., Gupta, A., Jahanshahi, R., Egele, M., & Joshi, A. (2018, May). Hardware performance counters can detect malware: Myth or fact?. In *Proceedings of the 2018 on Asia conference on computer and communications security* (pp. 457-468).

[10] Hunt, G., Letey, G., & Nightingale, E. (2017). The seven properties of highly secure devices. *tech. report* MSR-TR-2017-16.

[11] Yoon, M. K., Mohan, S., Choi, J., Kim, J. E., & Sha, L. (2013, April). SecureCore: A multicore-based intrusion detection architecture for real-time embedded systems. *In 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)* (pp. 21-32). IEEE.

[12] Sherwood, T., Perelman, E., Hamerly, G., Sair, S., & Calder, B. (2003). Discovering and exploiting program phases. *IEEE micro*, 23(6), 84-93. DOI: 10.1109/MM.2003.1261391

[13] Weicker, R. P. (1984). Dhrystone: a synthetic systems programming benchmark. *Communications of the ACM*, 27(10), 1013-1030.

[14] "Shakti: Open Source Processor Development Ecosystem, IIT Madras.."Available: https://shakti.org.in/.

[15] D. Patterson et al., "Embench: A Modern Embedded Benchmark Suite," 2019. Available: https://github.com/embench/embench-iot.

[16]"Coremark: An EEMBC Benchmark." https://www.eembc.org/.

[17]Godavarthi, S., & G., D. V. R. (2024). Federated Learning's Dynamic Defense Against Byzantine Attacks: Integrating SIFT-Wavelet and Differential Privacy for Byzantine Grade Levels Detection. *International Journal of Computational and Experimental Science and Engineering*, 10(4);775-786. https://doi.org/10.22399/ijcesen.538

[18]P. Jagdish Kumar, & S. Neduncheliyan. (2024). A novel optimized deep learning based intrusion detection framework for an IoT networks. *International Journal of Computational and Experimental Science and Engineering*, 10(4)1169-1180. https://doi.org/10.22399/ijcesen.597

[19]ONAY, M. Y. (2024). Secrecy Rate Maximization for Symbiotic Radio Network with Relay-Obstacle. *International Journal of Computational and Experimental Science and Engineering*, 10(3);381-387. https://doi.org/10.22399/ijcesen.413

[20]Jha, K., Sumit Srivastava, & Aruna Jain. (2024). A Novel Texture based Approach for Facial Liveness Detection and Authentication using Deep Learning Classifier. *International Journal of Computational and Experimental Science and Engineering*, 10(3);323-331. https://doi.org/10.22399/ijcesen.369

[21]S, P., & A, P. (2024). Secured Fog-Body-Torrent : A Hybrid Symmetric Cryptography with Multi-layer Feed Forward Networks Tuned Chaotic Maps for Physiological Data Transmission in Fog-BAN Environment. *International Journal of Computational and Experimental Science and Engineering*, 10(4);671-681. https://doi.org/10.22399/ijcesen.490

[22]R, U. M., P, R. S., Gokul Chandrasekaran, & K, M. (2024). Assessment of Cybersecurity Risks in Digital Twin Deployments in Smart Cities. *International Journal of Computational and Experimental Science and Engineering*, 10(4);695-700. https://doi.org/10.22399/ijcesen.494

[23]Prasada, P., & Prasad, D. S. (2024). Blockchain-Enhanced Machine Learning for Robust Detection of APT Injection Attacks in the Cyber-Physical Systems. *International Journal of Computational and Experimental Science and Engineering*, 10(4);799-810. https://doi.org/10.22399/ijcesen.539

[24]S, P. S., N. R., W. B., R, R. K., & S, K. (2024). Performance Evaluation of Predicting IoT Malicious Nodes Using Machine Learning Classification Algorithms. *International Journal of Computational and Experimental Science and Engineering*, 10(3);341-349. https://doi.org/10.22399/ijcesen.395

[25]C, A., K, S., N, N. S., & S, P. (2024). Secured Cyber-Internet Security in Intrusion Detection with Machine Learning Techniques. *International Journal of Computational and Experimental Science and Engineering*, 10(4);663-670. https://doi.org/10.22399/ijcesen.491

[26]guven, mesut. (2024). Dynamic Malware Analysis Using a Sandbox Environment, Network Traffic Logs, and Artificial Intelligence. *International Journal of Computational and Experimental Science and Engineering*, 10(3);480-490. https://doi.org/10.22399/ijcesen.460