A Hardware-Software Framework for IoT Security using Multi-Key Circuit Obfuscation

Alessandro Massaro[1], *Senior Member, IEEE*, Nicola Epicoco[1], *Senior Member, IEEE*, Giuseppe Loseto[1], *Senior Member, IEEE*, Filippo Gramegna[1] and Carmelo Antonio Ardito[2]

***Abstract*— This work proposes a framework for a hardware protection system aimed at enhancing malware detection in Internet of Things devices. The framework incorporates a circuit obfuscation mechanism based on the generation and management of cryptographic keys used to unlock and enable the functionality of the IoT hardware circuit. The proposed architecture relies on the design of a logic circuit capable of processing cryptographic keys generated and managed through Software Development Kit tools, thereby strengthening hardware-level protection against malware attacks. A proof-of-concept implementation of a hybrid hardware–software security system is presented, demonstrating the feasibility of an enhanced framework based on a multi-key architecture to improve system resilience and security.**

I. Introduction and Motivation

The wide adoption of Internet of Things (IoT) technologies in industrial environments has significantly increased the attack surface of Cyber-Physical Systems (CPSs)[1]. Industrial IoT (IIoT) platforms integrate heterogeneous hardware devices, embedded systems, and cloud-based services to support critical applications such as smart manufacturing, predictive maintenance, and industrial automation. However, the growing complexity and connectivity of these systems make them particularly vulnerable to cyber threats, including malware attacks targeting both software and hardware components [2]. Ensuring robust security mechanisms across the hardware–software stack has therefore become a critical requirement for protecting industrial infrastructures and sensitive operational data.

In IIoT environments, Software Development Kits (SDKs) provide tools and facilities for enabling the development, integration, and deployment of applications interacting with embedded devices and industrial control systems. SDKs often support access to low-level functionalities, communication interfaces, and hardware resources, making them a strategic component in the software stack of IIoT platforms. If improperly protected, SDKs may become a target for reverse engineering, code tampering, or information extraction by malicious applications and malware. Such vulnerabilities may expose sensitive implementation details or enable attackers to manipulate device behavior, potentially compromising the reliability and safety of industrial processes. Therefore, securing SDKs through mechanisms such as code obfuscation, cryptographic protection, and hardware-assisted security approaches is essential to ensure the integrity, confidentiality, and resilience of IIoT systems. SDKs, such as Studio 5000 by Rockwell Automation and similar industrial control programming platforms, are generally not exposed to untrusted applications; however, malware attacks may still extract information from the associated applications. The details of an SDK implementation can be hidden from the user through software obfuscation techniques that implement specific logic conditions. Additional threats may target hardware components directly, as in the case of Hardware Trojan (HT) attacks [3].

In this context, Electronic Digital Twins (EDTs) [4], based on circuit simulations, can support the understanding of hardware attack typologies by simulating undesired electrical coupling effects, such as resistive and capacitive interactions introduced by HT attacks [5], [6]. Circuit simulation can therefore serve as a useful tool for studying the interaction between hardware and software security mechanisms. EDTs are increasingly exploited in several application domains due to their capability to reproduce the behavior of physical electronic systems with high fidelity and reduced deployment costs. In robotic applications [7], EDTs enable the simulation and validation of control logic, sensor interactions, and actuator behavior before deployment on physical platforms. Similarly, EDTs can be also adopted in energy management systems [8] to enable the analysis of power consumption patterns, fault conditions, and optimization strategies in simulated environments, aiming to improve operational efficiency and resilience against cyber-physical threats.

Following the state of the art, this paper proposes a hardware-software solution integrating cryptographic key mechanisms within SDKs in order to protect hardware components from malware attacks. The proposed approach introduces a multi-key obfuscation framework that combines software-level protection with hardware-level verification through dedicated logic circuits. In particular, obfuscation is a security technique aimed at intentionally concealing the structure, logic, or behavior of software or hardware systems in order to make reverse engineering, unauthorized analysis, or malicious manipulation significantly more difficult while preserving the original functionality of the system. In this

This work was supported by the framework of the project "xTech NextHub: Competence Center for Innovative Solutions Development" through the Bando Regionale della Puglia per gli Aiuti in Esenzione 17 del 30/09/2014-BURP 139 suppl. del 06/10/2014 e s.m.i.-Titolo II Capo 2 del Regolamento Generale "Avviso per la Presentazione dei Progetti Promossi da Grandi Imprese ai Sensi Dell'articolo 17 del Regolamento."

[1]A. Massaro, N. Epicoco, G. Loseto and F. Gramegna are with the Department of Engineering, LUM "Giuseppe Degennaro" University, Strada Statale 100 km 18, Casamassima, 70010 Bari, Italy {massaro, epicoco, loseto, gramegna}@lum.it

[2]C. A. Ardito is with the Department of Electrical and Information Engineering, Polytechnic University of Bari, 70125 Bari, Italy. carmelo.ardito@poliba.it

architecture, the correct execution of SDK operations and the activation of IoT chip control signals are enabled only when specific key sequences are validated, preventing unauthorized access or manipulation by malicious software. By coupling SDK-based software control with hardware-assisted security mechanisms, the framework provides a layered protection strategy capable of strengthening the resilience of IoT devices against both software-based malware threats and hardware-oriented attacks. At the same time, the proposed methodology is not designed as a universal protection mechanism against all categories of cyberattacks. Vulnerabilities at higher software layers remain outside the direct protection scope of the presented framework. Instead, the contribution primarily focuses on securing embedded and IIoT-oriented CPSs, where attackers may attempt hardware tampering, firmware manipulation, unauthorized device interaction, or reverse engineering of industrial control logic.

The paper is structured as follows. The reference hybrid hardware–software framework is described in Section III. Section II illustrates the related work, whereas Section IV discusses the design of the logic circuit based on a two-key system in order to demonstrate the functionality of the proposed framework. A theoretical extension of the framework is presented in Section V, introducing a system based on the management of multiple cryptographic keys. Finally, the conclusions are presented in Section VI.

II. Related Work

Recent research has highlighted the importance of integrating software obfuscation with hardware-based protection mechanisms such as logic locking and circuit obfuscation. Logic locking techniques embed secret keys into hardware circuits so that correct functionality is achieved only when the appropriate key sequence is provided, thereby protecting intellectual property and preventing unauthorized circuit usage [9]. For example, a newly connected device would only be allowed to exchange sensitive data or execute privileged operations if it can provide the expected cryptographic sequence or obfuscated interaction pattern. This limits the possibility of unauthorized devices interacting with the IIoT infrastructure, even if the hardware or firmware is partially exposed. However, several studies have demonstrated that poorly designed obfuscation schemes may still expose structural vulnerabilities that can be exploited by advanced attacks, including machine-learning-based deobfuscation strategies [10]. For example, neural-network-based attacks such as BOCANet have shown the ability to analyze obfuscated circuits and recover hidden logic structures by learning patterns in circuit behavior [11]. Similarly, investigations into logic locking schemes have revealed in [12] that some security assumptions may be overly optimistic, as attackers may recover secret keys through side-channel analysis or structural inference methods if the obfuscation strategy is not sufficiently robust.

To address these challenges, hybrid hardware–software protection approaches have been proposed to combine code obfuscation at the software level with circuit-level obfuscation or hardware accelerator protection. Hardware–software co-design strategies allow security mechanisms to be distributed across multiple layers of the system architecture, increasing the overall difficulty for attackers attempting to bypass security controls [13]. In this context, the integration of software-level SDK protection with hardware-based key verification circuits can create a layered security framework in which both the software execution environment and the underlying hardware logic must be simultaneously compromised to break the protection mechanism.

Secure obfuscation techniques are increasingly studied as part of broader IoT security architectures that aim to detect and mitigate malware activity directly at the hardware level. For example, dedicated circuits capable of identifying malware signatures derived from processor-level information have been proposed to strengthen device-level protection mechanisms in IoT systems [14]. These developments demonstrate that combining SDK protection, circuit obfuscation, and hardware-based monitoring techniques can significantly enhance the security posture of modern IoT platforms.

Furthermore, Machine Learning (ML) techniques have been widely adopted to improve the detection and classification of malware in modern computing systems, including IoT environments [15]. ML models can analyze large volumes of behavioral and structural data extracted from software executions, enabling the identification of malicious patterns that are difficult to detect using traditional signature-based approaches. Several studies have explored the use of classification algorithms to identify malicious software from executable files and system behaviors. For instance, gradient boosting decision tree methods have been applied to static Portable Executable (PE) file analysis, demonstrating promising performance in malware detection scenarios [16]. Similarly, machine-learning-based antivirus tools relying on algorithms such as XGBoost have been developed to detect malicious software by analyzing feature sets derived from program execution and file structures [17].

More recent research has focused on dynamic malware analysis techniques that monitor runtime behavior and API interactions in order to improve classification accuracy and better capture evolving attack strategies [18]. In the context of IoT systems, deep learning approaches have also been proposed to analyze network traffic generated by connected devices, enabling effective malware detection through behavioral traffic analysis and multitask learning architectures [19]. These studies demonstrate that AI-based approaches can significantly enhance the capability of cybersecurity systems to detect previously unseen malware variants, providing a flexible and scalable solution for protecting software platforms and networked devices.

In this context, the proposed hardware-software obfuscation framework can complement ML-based malware detection techniques by providing hardware-level protection for IoT SDK operations, thereby creating a layered security architecture capable of mitigating both software and hardware-

oriented attacks.

III. Framework Architecture

The use of cryptographic keys within SDKs to protect hardware components from malware represents a promising approach to strengthening the overall security of IoT systems. IoT communication channels and chip control mechanisms are often exposed to cyber threats, and malware attacks may exploit software interfaces or hardware vulnerabilities to manipulate device behavior or extract sensitive information. In this context, obfuscation techniques can significantly increase the security level by introducing controlled access mechanisms across multiple layers of the system architecture.

In particular, the proposed framework introduces different protection mechanisms at the following levels:

- Key sequencer for IoT control encryption: a key-bit sequence is generated to lock the SDK application and prevent unauthorized access. The key generation process may be dynamic, further increasing protection against malware analysis and reverse engineering attempts;
- Software-level obfuscation: the SDK is protected through proprietary logic mechanisms that rely on the previously generated key sequence, ensuring that only authorized software components can access specific functionalities;
- Hardware-level obfuscation: the control circuit of the IoT chip can be unlocked only when the correct key-bit sequence is provided, preventing malicious software from directly controlling the hardware components.

Figure 1 illustrates the proposed hybrid hardware–software framework for IoT chip control, which integrates obfuscation and deobfuscation mechanisms to manage the secure locking and unlocking of the system.

In cybersecurity, obfuscation is a technique used to transform code or data into a form that is difficult to interpret or analyze while preserving its original functionality. The purpose of obfuscation is to prevent reverse engineering, unauthorized analysis, or tampering with software components. Conversely, deobfuscation refers to the process of reversing obfuscation by restoring intentionally scrambled, packed, or hidden code to a readable and analyzable form. Deobfuscation techniques are often employed in malware analysis and cybersecurity investigations to understand the behavior of protected or malicious software.

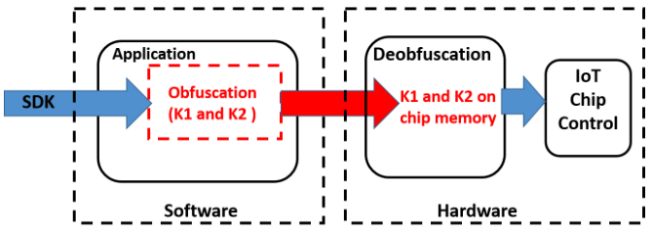


Fig. 1: Software–Hardware Co-Design Framework for IoT Control Encryption Based on Obfuscation Techniques

In IIoT environments, the inclusion of a hardware logic circuit significantly enhances security compared to relying solely on software obfuscation. Unlike software-only protections, the hardware verification mechanism is not easily bypassed or modified by malware, as it operates independently of the SDK code execution environment. This hardware layer provides tamper-resistant control, making it considerably more difficult for attackers to extract keys or manipulate the IoT device. Additionally, combining software and hardware security mechanisms enables multi-layered protection, increasing the resilience of the system against both software-based and hardware-targeted attacks.

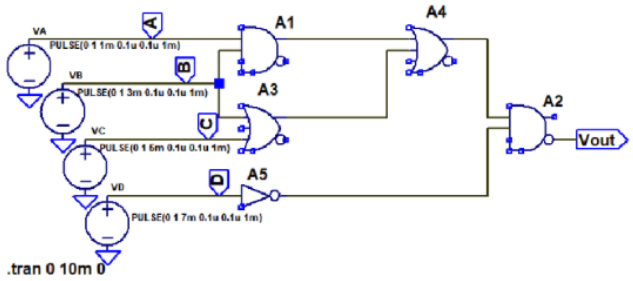
In particular, the co-design framework illustrated in Figure 1 is based on the following operational steps integrating both software and hardware components:

- SDK-based application execution: the application is executed through the SDK, which embeds security keys directly within the application code. In the proposed example, two cryptographic keys are considered; however, the framework can be extended to support multiple keys in order to increase the security level and system flexibility;
- Key verification through a hardware logic circuit: the software application containing the key combination interacts with a dedicated logic circuit responsible for verifying the correctness of the key sequence. When the correct key combination is recognized, the circuit enables the activation of the hardware control process;
- Secure signal transmission: once the key sequence has been successfully validated, the logic circuit authorizes the remote transmission of the signal processed by the SDK software, allowing the IoT device or chip to operate normally. If the key sequence is incorrect, the hardware control circuit remains locked, preventing unauthorized access or malicious manipulation of the system.

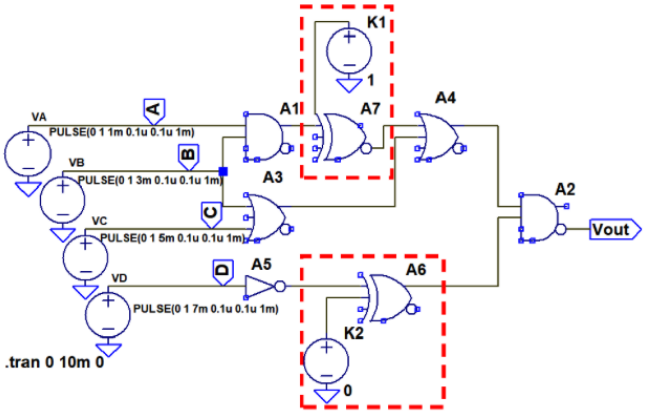
IV. Circuit Modeling

The hybrid software-hardware framework described in Section III provides a foundation for securing IoT devices through cryptographic key-based obfuscation and deobfuscation mechanisms. To illustrate the practical implementation of this framework, a basic example of IoT chip protection is presented in Fig. 2. The proposed circuit model was implemented using a SPICE-based approach with the open-source LTspice tool[1] (v.24.1.9 for x64 architectures), allowing precise simulation of voltage signals and logic behavior. Fig. 2a shows the unprotected circuit, which lacks any key-based access control, while Fig. 2b depicts the locked or protected circuit, incorporating a two-bit key sequence (K1 and K2) as a proof of concept. This reference example demonstrates the principle of integrating cryptographic keys directly into the hardware control logic, complementing the SDK-based software protection.

[1] https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html



(a) Example of logic circuit enabling an IoT communication based on pulse inputs (A, B, C, D)



(b) Obfuscation circuit adding two encryption keys (K1 and K2) protecting the transmitted signal

Fig. 2: Reference circuit modeling

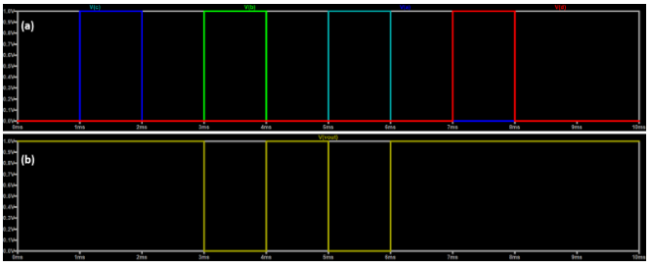


Fig. 3: Input pulses (a) at the input ports A, B, C, and D of the circuit of Fig. 2a and (b) related voltage output (Vout).

Fig. 3 illustrates an example of pulse excitation applied to the input ports of the logic circuit, simulating a sequence of bits corresponding to the IoT control signal and the output of the SDK software encoded by this bit sequence. By introducing obfuscation and deobfuscation techniques, the circuit is coupled with *XOR* and *XNOR* gates that enable or block operation depending on the specific combination of the keys K1 and K2, as highlighted by the red dashed lines in Fig. 3 (b). Only the correct key combination (K1 = 1, K2 = 0 in this example) allows the IoT control-coded signal to propagate, effectively enforcing hardware-level access control.

The voltage outputs of the encrypted circuit are plotted in Fig. 4, showing the behavior of the system under different key sequences. These results highlight the functionality and reliability of the hardware-based verification process in combination with the software-layer SDK protection. While the example uses a simple two-key system, the framework is readily extendable to a larger number of keys, increasing the system resilience against both software- and hardware-targeted attacks. The model also incorporates pulse signals consistent with typical IoT transmission protocols, where sequences of bits are represented as pulses. For practical hardware implementation, the logic gates shown in Fig. 2 can be realized using Bipolar Junction Transistors (BJTs) [20], enabling integration with real IoT devices.

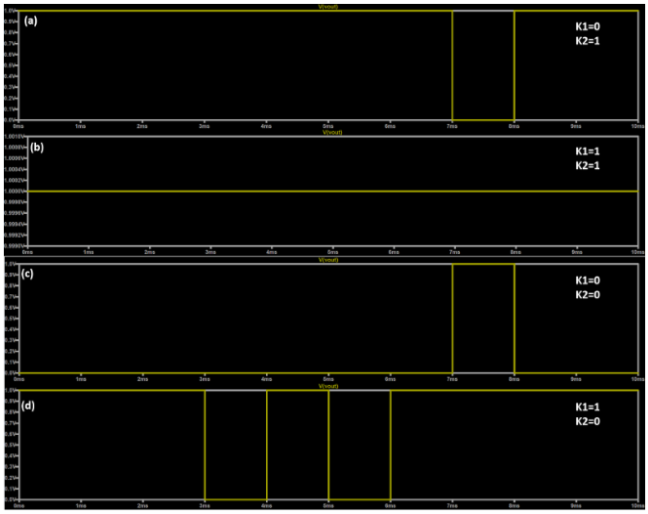


Fig. 4: Output voltage for different key combinations (K1, K2) with the same inputs as Fig. 2 (a). The correct combination, K1 = 1 and K2 = 0, reproduces the desired control signal shown in Fig. 2.

By connecting the SDK-based software control with the hardware key verification circuit, this approach demonstrates the effectiveness of a multi-layered security strategy, bridging the gap between software obfuscation and tamper-resistant hardware protection, as outlined in the framework section. The figures collectively illustrate the full workflow, from the unprotected IoT circuit (Fig. 2a to the locked hardware circuit (Fig. 2b), the pulse-based logic simulation (Fig. 3), and the corresponding voltage outputs under correct key activation (Fig. 4).

V. Multi-key Hardware Obfuscation

The logic circuit illustrated in Fig. 2b has been further extended in Fig. 5 to implement a four-key encryption mechanism, where only the correct combination K1 = 1, K2 = 0, K3 = 0, and K4 = 1 unlocks the output. This enhanced design significantly improves the security of the hybrid software–hardware system by increasing the complexity of key verification, making unauthorized access considerably more difficult. To further strengthen system resilience, the transmitted IoT signal can also be encrypted, providing additional protection against attacks targeting the output of the transmission channel.

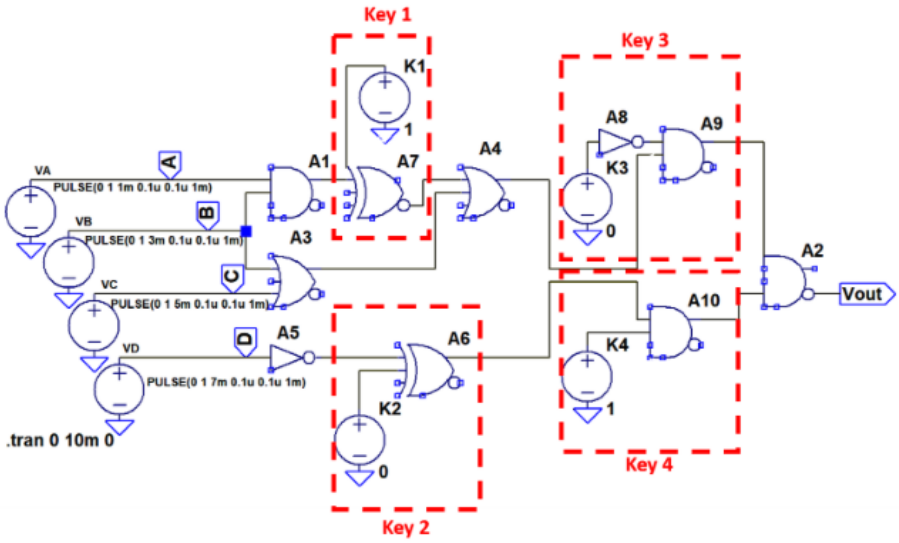


Fig. 5: Logic circuit based on four encryption keys.

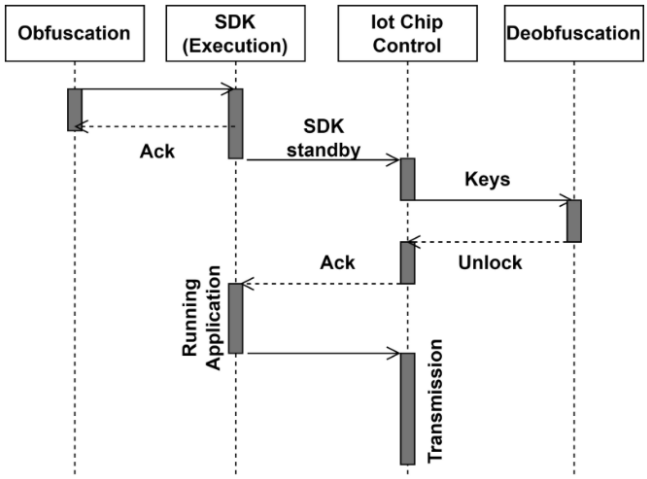


Fig. 6: UML sequence diagram illustrating the proposed operational workflow.

The overall operation and synchronization of the framework presented in Fig. 1 are represented using a Unified Modeling Language (UML) Sequence Diagram (SD) in Fig. 6. UML-SD is a standard graphical notation [21], [22] particularly suitable for modeling real-time interactions involving both software and hardware components [23]. By combining the four-key circuit design with the UML-SD workflow, the end-to-end process can be described as follows:

- Key-based obfuscation: The obfuscation process is activated using the four encryption keys embedded in the SDK;
- SDK standby: The SDK prepares to execute the application, remaining in a standby state while awaiting successful hardware deobfuscation;
- Hardware verification: The IoT chip unlocks only when the correct key combination is verified by the logic circuit, enforcing a secure hardware-level check;
- Application execution: Once the chip is unlocked, the SDK application is executed, maintaining alignment between software and hardware operations;
- Secure signal transmission: After a defined processing interval, the IoT chip transmits the output signal, which may be further encrypted to protect against external attacks.

This integrated description demonstrates how the proposed framework combines multi-key hardware obfuscation with software-layer control, creating a layered defense mechanism. By coordinating SDK execution with hardware verification, the system ensures that only authorized operations occur, effectively mitigating threats from both software-based malware and hardware-targeted attacks. The UML sequence diagram in Fig. 6 provides a clear visual representation of the timing and interaction of these processes, linking the four-key encryption mechanism with the software execution workflow in a real-time IoT scenario.

VI. Conclusion

The paper proposes a versatile framework for protecting SDK applications used in the control of IoT chips. The proposed approach introduces a multi-key encryption architecture that integrates both software and hardware components in order to strengthen the resilience of IoT systems against malware attacks. In particular, the framework combines software-level obfuscation mechanisms embedded in the SDK with hardware-level verification through a logic circuit capable of validating specific key sequences before enabling the execution of the application and the transmission of control signals. The functionality of the proposed architecture has been demonstrated through circuit simulations that model the deobfuscation process and the unlocking of the IoT chip based on the correct key combination. The proposed multi-key circuit approach can significantly increase the security level of IoT systems by introducing hardware-

assisted protection mechanisms that complement traditional software-based defenses. Furthermore, the framework can be extended toward a dynamic cybersecurity architecture by integrating cloud-edge intelligence techniques [24] and eXplainable Artificial Intelligence (XAI) approaches [25] capable of supporting real-time malware detection and adaptive security responses in IIoT environments.

Future developments will also include evaluations against standard attack methodologies, reverse engineering techniques, and other deobfuscation approaches. Additionally, we plan to investigate formal robustness analyses and security proofs in order to provide stronger theoretical guarantees regarding the resilience of the proposed mechanism. The definition of a structured evaluation framework is under consideration including quantitative security and performance metrics, benchmarking against existing obfuscation solutions, and systematic validation procedures. Finally, a sample application leveraging the SDK will be developed to demonstrate the proposed obfuscation method in a realistic real-world deployment scenario.

REFERENCES

[1] W. Fei, H. Ohno, and S. Sampalli, "A systematic review of IoT security: Research potential, challenges, and future directions," *ACM computing surveys*, vol. 56, no. 5, pp. 1–40, 2023.

[2] T. S. AlSalem, M. A. Almaiah, and A. Lutfi, "Cybersecurity risk analysis in the IoT: A systematic review," *Electronics*, vol. 12, no. 18, p. 3958, 2023.

[3] A. Massaro, N. Epicoco, G. Loseto, and C. Ardito, "IIOT Cyberattacks in Industrial Field Control Systems: PID Attack and Hardware Trojan Effects," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024, pp. 1702–1707.

[4] A. Massaro, "Electronic digital twin in industry 5.0: Ai-based electronic industry 5.0 models," in *AI-Driven Smart Industrial Technologies*. IGI Global Scientific Publishing, 2026, pp. 87–126.

[5] A. Massaro, N. Epicoco, G. Loseto, and C. A. Ardito, "AI-Based Detection and Simulation of Hardware Trojan Attacks in Industrial Applications," *IEEE Access*, vol. 13, pp. 50 967–50 979, 2025.

[6] A. Massaro, N. Epicoco, G. Loseto, G. Starace, and C. A. Ardito, "Smart Metering and Hardware Trojan Electronic Digital Twin based on Artificial Intelligence," *IFAC-PapersOnLine*, vol. 59, no. 9, pp. 247–252, 2025.

[7] A. Massaro and N. Epicoco, "Ai-assisted electronic digital twin for capacitive pressure sensors controlling robotic gripping," *IEEE Sensors Journal*, vol. 26, no. 9, pp. 13 338–13 346, 2026.

[8] G. Loseto, A. Massaro, and N. Epicoco, "Federated learning meets electronic digital twins for privacy-preserving hardware trojan detection in smart meter infrastructures," *IEEE Sensors Journal*, pp. 1–1, 2026.

[9] J. Gandhi, D. Shekhawat, M. Santosh, and J. G. Pandey, "Logic locking for IP security: A comprehensive analysis on challenges, techniques, and trends," *Computers & Security*, vol. 129, p. 103196, 2023.

[10] A. Darjani, N. Kavand, Z. Han, and A. Kumar, "Structurally Secure Obfuscation: Assessing and Mitigating Structural Vulnerabilities in Circuits Obfuscation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 31, no. 1, Nov. 2025. [Online]. Available: https://doi.org/10.1145/3772062

[11] F. Tehranipoor, N. Karimian, M. Mozaffari Kermani, and H. Mahmoodi, "Deep RNN-Oriented Paradigm Shift through BOCANet: Broken Obfuscated Circuit Attack," in *Proceedings of the 2019 Great Lakes Symposium on VLSI*, ser. GLSVLSI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 335–338.

[12] M. T. Rahman, S. Tajik, M. S. Rahman, M. Tehranipoor, and N. Asadizanjani, "The Key is Left under the Mat: On the Inappropriate Security Assumption of Logic Locking Schemes," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020, pp. 262–272.

[13] A. Chakraborty and A. Srivastava, "Hardware-Software Co-Design Based Obfuscation of Hardware Accelerators," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 547–552.

[14] Y. Matunaka, K. Tachihana, R. Kobayashi, and M. Kato, "A circuit for detecting IoT malware using signatures derived from processor information on LSI," *International Journal of Information Security*, vol. 24, no. 4, p. 175, 2025.

[15] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[16] H.-D. Pham, T. D. Le, and T. N. Vu, "Static PE malware detection using gradient boosting decision trees algorithm," in *International conference on future data and security engineering*. Springer, 2018, pp. 228–236.

[17] J. Palša, N. Ádám, J. Hurtuk, E. Chovancová, B. Madoš, M. Chovanec, and S. Kocan, "MLMD-A Malware-Detecting Antivirus Tool Based on the XGBoost Machine Learning Algorithm," *Applied Sciences*, vol. 12, no. 13, p. 6672, 2022.

[18] D. Z. Syeda and M. N. Asghar, "Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning," *Applied Sciences*, vol. 14, no. 3, p. 1015, 2024.

[19] S. Ali, O. Abusabha, F. Ali, M. Imran, and T. Abuhmed, "Effective multitask deep learning for iot malware detection and identification using behavioral traffic analysis," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1199–1209, 2022.

[20] M. K. Prasad, P. Bikkuri, and N. Manaswini, "Operation of logic gates (AND, NAND, OR, NOR) With single circuit using bjt (bipolar junction transistor)," *International journal of advance research ideas and innovations in technology (IJARIIT)*, vol. 5, no. 1, pp. 705–710, 2019.

[21] *ISO/IEC 19505-1:2012 – Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 1: Infrastructure*, International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) Standard, 2012, available from: https://www.iso.org/standard/58035.html.

[22] *ISO/IEC 19505-2:2012 – Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 2: Superstructure*, International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) Standard, 2012, available from: https://www.iso.org/standard/58036.html.

[23] X. Li and J. Lilius, "Checking compositions of UML sequence diagrams for timing inconsistency," in *Proceedings Seventh Asia-Pacific Software Engeering Conference. APSEC 2000*. IEEE, 2000, pp. 154–161.

[24] D. Loconte, S. Ieva, F. Gramegna, I. Bilenchi, C. Fasciano, A. Pinto, G. Loseto, F. Scioscia, M. Ruta, and E. Di Sciascio, "Serverless microservice architecture for cloud-edge intelligence in sensor networks," *IEEE Sensors Journal*, vol. 25, no. 5, pp. 7875–7885, 2024.

[25] D. Loconte, S. Ieva, G. Mascellaro, A. Pinto, G. Loseto, F. Scioscia, and M. Ruta, "On-Device Explainable Artificial Intelligence for the Semantic Web of Everything," *Future Generation Computer Systems*, p. 108310, 2025.