Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 2-A, pp. 397–410
DOI : 10.47974/JDMSC-2470

Implementation of IRIS authentication on industrial IoT devices for secured boot process

Pragya Vaishnav[@]
Department of Computer Applications
Manipal University Jaipur
Jaipur 303007
Rajasthan
India

Md. Ashraful Babu [†]
Department of Physical Sciences
Independent University
Dhaka 1229
Bangladesh

Swati Jain [§]
Department of Commerce
Manipal University Jaipur
Jaipur 303007
Rajasthan
India

Shilpa Sharma [‡]
Vaibhav Bhatnagar *
Department of Computer Applications
Manipal University Jaipur
Jaipur 303007
Rajasthan
India

[@] *E-mail:* pragya.vaishnav@jaipur.manipal.edu
[†] *E-mail:* ashraful388@gmail.com
[§] *E-mail:* swati.jain@jaipur.manipal.edu
[‡] *E-mail:* shilpa.sharma@jaipur.manipal.edu
* *E-mail:* vaibhav.bhatnagar@jaipur.manipal.edu (Corresponding Author)

© Taru Publications

Abstract

Instant Retrieval Information System (IRIS) is a hardware subsystem designed to provide IoT devices with secure boot functionality. The author presents an iris can be used to IoT devices, together with a hardware secure boot solution. IRIS has the ability to boot a Linux kernel image that has been pre-stored on removable media. It also has a data validator that ensures the boot process's secrecy, integrity, and validity. It has been observed that with the use of field programmable gate array chips, IRIS has short boot up times and a tiny hardware footprint. Subsequently, Electronic Linux Unified Key Setup (Embedded LUKS) an open-source crypto-core, can be implemented outside of the boot loading process to add secrecy, integrity, and validity to data saved on off-chip storage, such as a flash device. Thus, IRIS demonstrated an open-source generic solution that can be tailored to many architectures.

Subject Classification: *68M25*

Keywords: *Industrial IoT, IRIS, Boot, Security, Verification, Chain of trust (CoT), Unified extensible firmware (UEFI).*

1. Introduction

Secure Boot is essential to strengthening IoT devices in today's networked environment. By verifying digital signatures, it protects boot-time software from malicious assaults and guarantees its integrity and reliability. This crucial security feature verifies the authenticity of boot-loaded software components and is integrated into device firmware, such as BIOS or UEFI. Notably, safe Boot creates a safe foundation for IoT systems by lessening vulnerability to firmware-level threats and rootkits [1]. This is a revie paper of IRIS: An embedded secure boot for IoT devices [1].

In order to guarantee the validity and integrity of the firmware and software on Internet of Things devices, secure boot is a crucial security feature. Security boots are essential for IoT devices for the following reasons:

Preventing illegal software: On Internet of Things devices, secure boot helps stop malicious or unauthorized software from running. It ensures that only duly signed and reliable software is loaded by confirming the integrity and validity of the software during bootup.

Preventing firmware tampering: The software that governs the operation of many IoT devices is frequently modified. Secure boot guarantees that unauthorized users haven't altered or tampered with the firmware [2]. It stops the device from booting up if it finds any unapproved alterations, including the introduction of malware or malicious code.

Defending against supply chain attacks: Secure boot reduces the possibility of supply chain attacks, in which hackers try to infiltrate

devices while they are being manufactured or distributed. Secure boot helps guarantee that only reliable parts are in the device by confirming the integrity and validity of the firmware and software at boot time.

Improving overall security system: One fundamental security measure that fortifies the overall security posture of Internet of Things devices is secure boot. It helps create a chain of trust for upcoming software and firmware updates and offers a reliable foundation for the device's functionality[3]. It decreases the attack surface and increases resistance to different security threats by making sure that only authorized and properly signed code operates on the device.

Based on the underlying hardware, a large variety of IoT devices may be roughly differentiated into two parts: system on chips and microcontroller units [4]. While both have a bootloader, the software that is executed in each scenario varies since bootloader often launches standalone software, whereas the operating system is started in the later scenario. In the last case, the bootloader's goal is to load the operating system kernel; but, due to its size, this is frequently not achievable in a single step. In this instance, dividing the boot loading procedure into several software phases and establishing a bootloader chain is the chosen course of action. Usually referred to as the pre-loader, the first stage begins when the SoC initializes a minimal hardware configuration [5].

The objective of this research is to review a secure boot solution for IoT devices through the Instant Retrieval Information System (IRIS) and increase the efficiency of boot time for IoT devices. In this paper Section 2 presents the literature reviews of previous studies on IRIS for IoT devices and section 3 presents the proposed system architecture. Section 4 presents IRIS aims for secured boot process, Section 5 presents the designed algorithm and Then, Section 6 presents the result of the proposed system and Section 7 Compares the Boot Process with IRIS and without IRIS and then Section 8 describe the secure boot challenges. Lastly, the conclusion is drawn in section 9.

2. Literature Review

A number of security solutions are currently available with features such as integrity, privacy, and authentication [5]. These three signs can help prevent further attacks. In order to verify the provenance of data and ensure that it has not been tampered with by someone else, for example, accuracy and integrity are required. These two functions can prevent attacks on firmware components as described in [6]. Adding confidential

changes or confidential information to the kernel image or source data that needs to be protected is a common practice. By encrypting the data, reverse engineering attacks on the firmware originating from the device can be prevented [7]. To support integrity, privacy, and authenticity, a full hardware-secure boot using a programmable system-on-a-chip (PSoC) is proposed in [8]. This architecture consists of two parts: the formal logic area (PL area), which contains the FPGA, and the processing system area (PS area), which contains the ARM processor. Secure boot, which uses factory-generated FPGA unique IDs and non-volatile memory (NVM) for authentication, is implemented by the PL circuit. The image is stored using symmetric encryption to ensure confidentiality and all data is processed using a hash to ensure integrity. However, the encryption methods provided, such as AES, are not the best option for mid-range and low-end devices, as the FPGA demonstrates.

ITUS, a multi-module, fully secure system proposed in [5], is another suitable answer to this question. The hardware core verifies and signs every step in the IoT implementation module for secure execution. ITUS is a medium-scale solution tested on various FPGA devices, as its main goal is to protect against quantum computing vulnerabilities [9]. Therefore, it uses techniques such as elliptic curve digital signature or the extended Merkle signature scheme to increase integrity; however, these algorithms are not suitable for constrained vascular devices, which are commonly seen in the Internet of Things industry. Although the design first demonstrated by CARE in [10] combines hardware and software. The CARE design provides reliability and integrity by incorporating a recovery mechanism that triggers a pre-stored secure image. The hardware unit contains a data verifier that ensures the integrity and validity of the image stored in the flash memory. The verification method splits the image into two halves and includes a signed digest [11]. The module then analyzes each segment of the flash image independently. Furthermore, since only HMAC modules are implemented in hardware, the CARE architecture is not a complete hardware design and does not provide data protection.

3. System Architecture

IRIS is a hardware subsystem that aims to give IoT devices secure boot capabilities. It can be included into a device to give the boot process confidentiality, integrity, and authenticity. IRIS's primary characteristics are:

- It is an open-source licensing that facilitates simpler integration and reusability in other projects [6].
- It is an easy solution to modify various CPUs and bus types, and it provides a general secured boot for Embedded systems [7].
- This is a small hardware footprint that makes it ideal for devices with limited resources.

Only a small number of the IRIS subsystem's components, with the internal processor bus need to be modified when it is integrated into a system on a chip (SoC). Consequently, IRIS subsystem will take over the initial boot process and load a Linux kernel image into RAM, decrypt, and verify it, allowing the CPU to execute it only if the verification step is successful [8]. It will retrieve the kernel image from a storage device to store the encrypted Linux kernel image on external media or off-chip flash memory.

IRIS is made up of several parts and a modular design. The Embedded LUKS Module and the Boot module are the two primary components of IRIS. The Embedded LUKS Bridge connects the Embedded LUKS Module to the bus, which may need to be modified to serve as the Boot module's internal interface to the bus, depending on the system bus that the SoC employs [9]. To enable IRIS to boot a Linux kernel that is saved in the storage device, all modules (Embedded LUKS and Boot module) must rewrite its connection to the system bus. An SD host reader module can access an SD card as storage media in the current iteration of IRIS.

The Boot module is the primarily manages the system's boot process by seizing control of every component linked to the internal bus of the

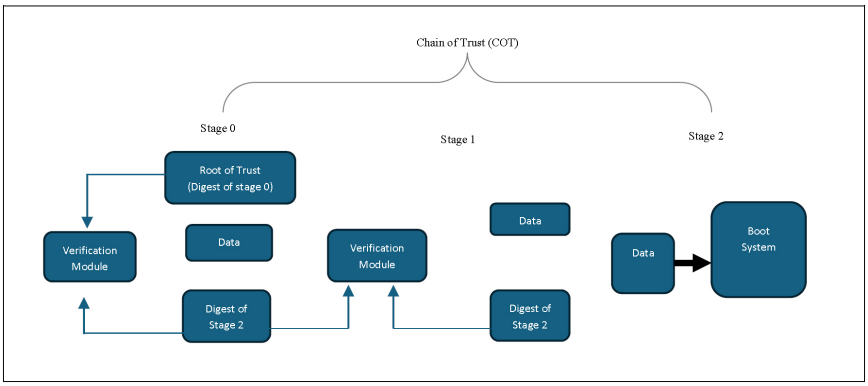


Figure 1

The secure boot process

processor. The Embedded LUKS Module facilitates information verification & security. By using encryption, the Linux kernel image kept in the external memory (SD card), confidentiality is achieved. The output of a Hash-based Message Authentication Code (HMAC) function, that is encrypted with the same key as the data, is attached to these encrypted files. By preventing data manipulation, creation, or corruption during the boot-up process, this guarantees data integrity and validity.[10]. The IRIS boot up procedure will be explained in the upcoming section followed by the details of IRIS's, Boot module and Embedded LUKS in subsequent section.

A series of steps make up the secure boot process, where each step validates the one before it is launched. Fig. 1 provides a broad picture. The element known as the Root of Trust (RoT), which is typically an immutable pre-loader kept in an on-chip ROM, is regarded as trustworthy by default. A summary of the data from each stage is attached and will be utilized in an initial verification process.

Only integrity and authentication are supported by secure boot implementations like those found in UEFI; however, when the Linux kernel is used, confidentiality can be added to the boot process by utilizing the LUKS [1,2]. Thus, IRIS, a proprietary secure boot hardware for IoT devices, is proposed in this contribution. It is designed to enable low-footprint SoC devices to safely boot a Linux operating system. Encryption, integrity, and authentication are guaranteed by IRIS while booting an off-chip memory-stored Linux kernel image.

IRIS for IoT Devices to secure Boot Process:

IRIS is a hardware subsystem that can be included into Internet of Things devices to enhance the boot process' secure boot features. Its purpose is to give the boot process confidentiality, integrity, and authenticity. Among its attributes are:

- Small hardware footprint: IRIS works well with devices with limited resources.
- Generic solution: For Embedded systems, IRIS is a generic secure boot solution. [13].
- Open-source licensing: IRIS can be more easily integrated and reused in other applications because of its open-source license.
- Flexible: IRIS is adaptable to many buses and CPUs.

Secure boot in IoT devices

- Secure Boots makes sure that only reliable and duly signed software components are run during the boot process, guarding against the execution of unauthorized code [14]. By doing this, you can protect the device from potential security risks by preventing it from running dangerous or unapproved code.
- Secure boot protects the integrity of the system by confirming that crucial software and the device's firmware haven't been altered or tampered with. This lowers the possibility of compromised or unreliable operations and increases the integrity of the system as a whole.
- Secure boot guards against software manipulation or substitution during manufacture, distribution, or deployment by authenticating software components throughout the boot process, thereby reducing supply chain risks. This guarantees that the gadget runs on authentic and reliable software.
- Furthermore, by functioning as a defense mechanism and blocking the execution of illegal or harmful code, secure boot contributes to a decrease in malware infestations.[15]. It lessens the chance of infection and guards against possible compromise or damage to the device and its data.

4. IRIS Process

A hard reset or SoC power-on initiates the IRIS secure boot process. To seize control of the internal bus and hold it there until the secure boot is finished, the Boot module first asserts the processor's reset signal. Secret parameters are needed during the boot process to initialize the Embedded LUKS Module. The Boot module, which oversees starting the Embedded LUKS Module, has them preloaded. The specifications are as follows:

PSW: a user password with a largest size of 80 bits that is used to decrypt and validate the Embedded LUKS boot partition.

Boot module: The Boot module is a complete hardware module that has been designed to manage the boot process. This module serves as a core component that enables or disables the many participating modules in the boot process; it is independent of the CPU. It connects to the other components via the bus system and has a control signal (cpu_reset) for resetting the CPU. This module's finite state machine (FSM) is its central component; it gradually establishes the order in which the other modules activate.

The Boot module first awaits a start signal. As the module is turned on, the main processor is stopped, and its reset is turned on. This reset is kept on until the Boot module is done with its work. After seizing control, the Boot module starts sending the necessary signals to the Embedded LUKS Module, including the encrypted partition password, whether to activate the HMAC function of E- LUKS, and where on the SD card the Embedded LUKS boot partition should start. Following the transmission of all data via the system bus, the Embedded LUKS Module returns decrypted data segments that the Boot module demands from the boot partition [4]. The Boot module saves the data in the RAM after it gets there[16]. Until all the data has been transferred to the RAM, this request procedure keeps going. The boot process is now complete as the Boot module deactivates the processor's reset.

E- LUKS: A complete hardware module called Embedded LUKS facilitates the data security procedure in the IRIS secured boot system. This functions similarly to LUKS and was primarily created for Internet of Things devices [12]. But Embedded LUKS is a hardware implementation whereas LUKS is a software driver.

5. Algorithm for Boot module for subsystem

IRIS has been adapted as a SoC on top of Digilent's Nexys4-DDR design, which incorporates the Xilinx Artix7 XC7A100T-1CSG324C FPGA. The machine inserts the SD card into the bootable media and writes a 6.3 MB Linux image to the E-LUKS partition.

1. BEGIN
2. Idle
3. WHILE system is idle
4. WAIT for start signal
5. END WHILE
6. Activate CPU_rst // System starts
7. SendPasswordToELUKS() // Send password to EMBEDDED LUKS module through system bus
8. SendHmacEnableToELUKS() // Send Hmac enable to EMBEDDED LUKS module through system bus
9. SendStartDirectionToELUKS() // Send start direction to boot partition to EMBEDDED LUKS through system bus

10. DecryptedData = RequestDecryptedDataFromELUKS() // Request data decrypted through EMBEDDED LUKS
11. Store // Store decrypted data to RAM through system bus

As a result, the author achieved the utilization of IRIS resources and boot time performance are analyzed. Embedded LUKS offers user data saved in a partition secrecy, integrity, and validity. The Embedded LUKS header on this partition, which contains details on the public cryptographic parameters utilized, is included in an unencrypted area. Additionally, it keeps slots for various users, all of them has a distinct password generated from the master key that is implemented to validate and encrypt user data. [17-18]. This master key's digest is contrasted with the original master key's digest that was kept inside the partition. The validity and integrity of the user data are examined in case the candidate is accurate. The user can fetch the data if all of it is accurate. Embedded LUKS differs from the original LUKS in a few significant ways, the most important of which are: It is designed to be readily incorporated as a component of a SoC.

- Embedded LUKS consists of three cryptographic algorithms—PRESENT, SPONGENT, and a variant of the key derivation function (KDF) first described in that are especially tailored for Internet of Things devices with constrained resources.
- The parameter count is employed to keep track of the no of iterations required before the KDF produces an output. The KDF function requires three parameters. You can change this count option to strengthen defenses against brute-force attacks. The security that is attained in this manner is thought to be equal to a key length of 80 + log2($colt$).
- To calculate the HMAC of the complete encrypted user data, Embedded LUKS adds a new process. Data integrity and authentication are guaranteed by this operation, both of which are necessary for a secure boot.
- Embedded LUKS optimizes LUKS internal data structures by matching the physical layer's logic block size and minimizing the number of internal fields. To improve performance on SD cards, the block size in the current implementation is 512 bytes.

6. Results

This is review for IRIS: An embedded secure boot for IoT devices [1]. First, boot time performance and IRIS resource use are examined. Secondly,

the way in which IRIS basic resources are utilized is contrasted with the other methods that are presented.

IRIS resource usage and boot time efficiency: IRIS tried in a SoC installed on the Diligent Nexys4-DDR prototype board, that features a Xilinx Artix7 XC7A100T-1CSG324C FPGA chip, in order to validate the suggested solution [9]. As boot media, the system comes with a removable SD card that has an E Embedded LUKS partition with a 6.3 MB encrypted Embedded Linux image on it.

In both implementations, the SD host and Boot modules require about the same resources. The slight discrepancy results from minor differences in the modules' interfaces when the Embedded LUKS Module is not in operation. Because of the inherent complexity of the cryptographic algorithms used by the Embedded LUKS Module, it uses the majority of the resources in the IRIS system (between 80% and 90% of all resources used). However, the IRIS solution is fully functional for the platform utilized in this SoC, with resource consumption being less than 10% of the device's available capacity.

As a result, the increase in bits is only significant if brute-force attacks are the most effective way to attack the system, as is the case with present, one of the algorithms suggested for Internet of Things applications [14].

7. Comparison of Boot Process with IRIS and without IRIS

IRIS is a multi-architecture-adaptable hardware secure boot solution for Internet of Things devices. This open-source solution comes with a crypto-core known as Embedded LUKS that may be used to enhance data saved on off-chip storage in terms of authenticity, integrity, and confidentiality. The comparison of the booth process with and without IRIS is shown in Table 1 [18].

Table 1

Comparison of Boot Process without IRIS and with IRIS

S. No	Boot Process without IRIS	Boot Process with IRIS
1	A key security feature that verifies the integrity of a device's firmware and software	IRIS gives the boot process secrecy, integrity, and authenticity. After the boot process, it is still accessible during runtime and can be used to read encrypted data on removable media[19].

Contd...

2	**Initialization:** When a device starts, the firmware checks the PK.	**Initialization:** A hard reset or SoC power-on initiates the IRIS secure boot process. To seize control of the internal bus and hold it there until the secure boot is finished, the Boot module first asserts the processor's reset signal[20]. Secret parameters are needed during the boot process to initialize the Embedded LUKS Module.
3	**Verification:** The firmware verifies the bootloader against the KEK and signature databases.	**Verification:** IRIS subsystem will take over the initial boot process and load a Linux kernel image into RAM, decrypt, and verify it.
4	**Execution:** If the verification is successful, the bootloader is executed. If not, the boot process is stopped.	**Execution:** allowing the CPU to execute it only in case if verification step is successful. It will retrieve the kernel image from a storage device to store the encrypted Linux kernel image on external media or off-chip flash memory.

8. Secure Boot Challenges

The device's root keys, which are used to generate a distinct device identification certificate, form the basis of the Secure Boot procedure. On device key generation (ODKG) should be used to produce a keypair within the device during device provisioning. Subsequently, the certificate authority (CA) receives a certificate signing request (CSR) in order to generate a signed certificate that is subsequently placed in the device. The Secure Boot procedure relies heavily on the security of these device keys.

Maintaining the security of these keys is therefore the biggest difficulty. The boot image is compared to the device processor's stored key. The boot image is launched if those two match. The chain of roots that starts the IoT device's operation is made up of matches to the root key in the CPU. The entire Secure Boot procedure is vulnerable if the keys are compromised.

Every device has a unique private key, which is never removed from the edge device. For the duration of the IoT device, these private keys can be generated again when the device certificate expires. Users of Internet of Things devices should also be aware that the Secure Boot procedure does not completely lock down the system. Only the operating system software is protected. This is crucial because a malware

that operates on top of the operating system can still compromise a device if it is inserted.

9. Conclusion

This article review the IRIS system, a low-footprint FPGA-based complete safe hardware boot solution for IoT devices. IRIS gives the boot process secrecy, integrity, and validity. After the boot process, it is still accessible during runtime and used to read encrypted data on removable media. IRIS is an open-source design that may be modified to fit many SoC architectures by changing the internal bus' glue logic.

Some modifications are currently being developed to address potential issues with this document as future work. Adding other cryptographic algorithms to IRIS, which are presently under study, is one of these upgrades. In particular, block ciphers LEA and CLEFIA, should be added. Furthermore, a way for the system to recover if data is tampered with, like setting up a central server where users may connect and upload a secure kernel image. To enable safe connection with the server, asymmetric cryptography must be implemented in order to construct this central server. Furthermore, it's critical to make sure that there is a mechanism in place for each FPGA to receive its password to boost system security.

References

[1] G. Cano-Quiveu, P. Ruiz-de-Clavijo-Vazquez, M. J. Bellido, J. Juan-Chico, and J. Viejo-Cortes, "IRIS: An embedded secure boot for IoT devices," *Internet of Things*, vol. 23, pp. 100874 (Jul. 2023), doi: https://doi.org/10.1016/j.iot.2023.100874.

[2] S. Umer, A. Sardar, Ranjeet Kumar Rout, M. Tanveer, and I. Razzak, "IoT-Enabled Multimodal Biometric Recognition System in Secure Environment," *IEEE internet of things journal* (Online), vol. 10, no. 24, pp. 21457–21466 (Dec. 2023), doi: https://doi.org/10.1109/jiot.2023.3299465.

[3] P. Vaishnav, M. Kaushik, and L. Raja, "Behavioral biometric authentication on smartphone using keystroke dynamics," *Journal of Discrete Mathematical Sciences & Cryptography/Journal of discrete mathematical sciences & cryptography*, vol. 26, no. 2, pp. 591–600 (Jan. 2023), doi: https://doi.org/10.47974/jdmsc-1645.

[4] J. Zhang, Y. Zhou, R. Xi, S. Li, J. Guo, and Y. He, "Iris," Proceedings of the ACM on Interactive Mobile Wearable and Ubiquitous Technologies, vol. 7, no. 3, pp. 1–27 (Sep. 2023), doi: https://doi.org/10.1145/3610913.

[5] P. Vaishnav, L. Raja, P. Singh, and Ramakrishnan Vairavasamy, "Multilayered authentication for ATM transaction using keystroke dynamics and touch dynamics," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 4, pp. 1357–1365 (Jan. 2024), doi: https://doi.org/10.47974/jdmsc-1990.

[6] M. B. Mohammad, Prudhvi Kanth Bezawada, Harish Tanneeru, and Jitendra Janjanam, "Development of standalone iris recognition system using deep neural networks," AIP conference proceedings, vol. 2931, pp. 090006–090006 (Jan. 2023), doi: https://doi.org/10.1063/5.0178605.

[7] S. Phani Praveen, Sai Srinivas Vellela, and B. Ramachandran, "SmartIris ML: Harnessing Machine Learning for Enhanced Multi-Biometric Authentication," *ResearchGate*, vol. 4, no. 1, pp. 25–36 (Jan. 2024), Accessed: May 14, 2025. [Online]. Available: https://www.researchgate.net/publication/378439449_SmartIris_ML_Harnessing_Machine_Learning_for_Enhanced_Multi-Biometric_Authentication

[8] Muhammad Imran Zulfiqar and I. Younis, "Enhanced Security Paradigms: Converging IoT and Biometrics for Advanced Locker Protection," *IEEE Internet of Things Journal*, pp. 1–1 (Jan. 2024), doi: https://doi.org/10.1109/jiot.2024.3432282.

[9] Omar Ibrahim Obaid and Saba Abdul-Baqi Salman, "Security and Privacy in IoT-based Healthcare Systems: A Review," *Mesopotamian Journal of Computer Science*, pp. 29–40 (Dec. 2022), doi: https://doi.org/10.58496/mjcsc/2022/007.

[10] P. Vaishnav, L. Raja, P. Singh, and S. Tandel, "Applications of artificial intelligence in cyber security," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 4, pp. 1367–1375 (Jan. 2024), doi: https://doi.org/10.47974/jdmsc-1991.

[11] K. Saminathan, T. Chakravarthy, and M. Chithra, "Iris Recognition Based On Kernels Of Support Vector Machine," Online) Ictact Journal On Soft Computing: Special Issue On Soft -Computing Theory, Application And Implications In Engineering And Technology, pp. 2 (2015), doi: https://doi.org/10.21917/ijsc.2015.0125.

[12] K. Nguyen, H. Proença, and F. Alonso-Fernandez, "Deep Learning for Iris Recognition: A Survey," *ACM computing surveys*, vol. 56, no. 9, pp. 1–35 (Apr. 2024), doi: https://doi.org/10.1145/3651306.

[13] S. Sharma, L. Raja, V. Bhatnagar, D. Sharma, Swami Nisha Bhagirath, and Ramesh Chandra Poonia, "Hybrid HOG-SVM encrypted face detection and recognition model," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 25, no. 1, pp. 205–218 (Jan. 2022), doi: https://doi.org/10.1080/09720529.2021.2014141.

[14] P. Patel, "How Secure Boot help to Secure IoT Device," eInfochips, Nov. 24 (2023). https://www.einfochips.com/blog/how-secure-boot-help-to-secure-IoT-device (accessed May 14, 2025).

[15] K. Bhateja, S. Sharma, S. Chaudhury, and N. Agrawal, "Iris recognition based on sparse representation and k-nearest subspace with genetic algorithm," *Pattern Recognition Letters,* vol. 73, pp. 13–18 (Dec. 2015), doi: https://doi.org/10.1016/j.patrec.2015.12.009.

[16] S. Umer, B. C. Dhara, and B. Chanda, "A novel cancelable iris recognition system based on feature learning techniques," *Information Sciences,* vol. 406–407, pp. 102–118 (Sep. 2017), doi: https://doi.org/10.1016/j.ins.2017.04.026.

[17] A. A. El-Latif, B. Abd-El-Atty, M. S. Hossain, S. Elmougy, and A. Ghoneim, "Secure Quantum Steganography Protocol for Fog Cloud Internet of Things," *IEEE Access*, vol. 6, pp. 10332–10340 (2018), doi: https://doi.org/10.1109/access.2018.2799879.

[18] P. Dash, F. Pandey, M. Sarma, and D. Samanta, "Efficient private key generation from iris data for privacy and security applications," *Journal of Information Security and Applications*, vol. 75, pp. 103506 (May 2023), doi: https://doi.org/10.1016/j.jisa.2023.103506.

[19] Kumar, P. Dadheech, V. Singh, L. Raja, and R. C. Poonia, "An enhanced quantum key distribution protocol for security authentication," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 22, no. 4, pp. 499–507 (May 2019), doi: https://doi.org/10.1080/09720529.2019.1637154.

[20] K. Hajari, U. Gawande, and Y. Golhar, "Neural Network Approach to Iris Recognition in Noisy Environment," *Procedia Computer Science,* vol. 78, pp. 675–682 (2016), doi: https://doi.org/10.1016/j.procs.2016.02.116.

Received April, 2025