# A PUF-Based Secure Boot for RISC-V Architectures

Stefano Di Matteo(✉), Luca Zulberti, Federico Cosimo Lapenna, Pietro Nannipieri, Luca Crocetti, Luca Fanucci, and Sergio Saponara

Department of Information Engineering, University of Pisa, Via G. Caruso 16, Pisa, Italy
stefano.dimatteo@dii.unipi.it

**Abstract.** Recently, there has been a growing interest in Physically Unclonable Functions (PUFs). These electronic circuits possess several key characteristics such as unpredictability and uniqueness that make them particularly attractive for security applications. PUFs offer an appealing solution for secure boot applications, providing a hardware-based mechanism for generating unique cryptographic keys. These keys can be used to encrypt the bootloader and operating system, thereby enhancing security. In this paper, we propose an innovative, secure boot scheme that leverages the functionality and characteristics of a PUF. Our approach eliminates the need for physical storage of the encryption key of the boot code, which enhances security and provides the possibility of securely updating the firmware. We will present an architecture that comprises essential components, along with a demo board on FPGA. The demo board features a general-purpose 64-bit RISC-V-based system that leverages the proposed PUF-based secure architecture, enabling secure boot and firmware update functionalities.

**Keywords:** PUF · RISC-V · Secure boot · Hardware · FPGA

## 1 Introduction

In today's digital age, security has become an increasingly critical concern for individuals, organizations, and governments. With the growing complexity and sophistication of cyber threats, it is crucial to implement robust security measures to protect computer systems and sensitive data. Secure boot technology is one of such security measure that is integrated into modern computer systems [5,7,10]. Its primary aim is to verify the digital signatures of the bootloader and operating system before loading, ensuring that only trusted software is executed during the boot process. This is achieved by creating a chain of trust that begins with the system firmware and continues through the bootloader and operating system [8]. If any of these components are compromised, the secure boot process will fail, and the system will not boot, ensuring that only reliable software is executed.


F. Bellotti et al. (Eds.): ApplePies 2023, LNEE 1110, pp. 87–94, 2024.
https://doi.org/10.1007/978-3-031-48121-5_13

Dedicated hardware can adopted to enhance the security of the chain of trust [8,9] and the performance of the entire system [2,3]. Physically Unclonable Functions (PUFs) are electronic circuits that have gained increasing interest in recent years due to their unique properties, making them ideal for security applications [11,12,14]. PUFs can generate unpredictable and unique responses to input stimuli, which is advantageous for creating cryptographic keys that are difficult to replicate or guess. The most attractive properties of a PUF, in fact, are the unpredictability of its response and the uniqueness of the response to each instance of the circuit. These properties result from the inherent variations in the manufacturing process fabricating the circuit. The response generated by the PUF can serve as a cryptographic key for encrypting the boot code, firmware, and operating system. Furthermore, another advantage of using the PUF response as a key is that it eliminates the need to preserve the secret key within the device.

Our paper presents a novel approach for enhancing security during boot-up, utilizing a security subsystem that leverages the functionality and unique characteristics of a PUF. By implementing this approach, we aim to address certain security concerns associated with secure boot, while also enabling secure firmware updates when necessary or in the event of a successful attack on a chip. This is crucial in preventing potential security breaches that could impact other chips of the same type. The paper is organized as follows: Chap. 2 explains our PUF-based secure boot concept and the hardware components needed to implement it. Chapter 3 presents a complete RISC-V-based system prototype composed of a general-purpose subsystem enhanced by a secure subsystem equipped with an FPGA-based Ring-Oscillator PUF to perform secure boot and secure firmware updates. Chapter 4 will draw the conclusions of this work.

## 2 Proposed PUF-Based Secure Boot

One of the classical approaches to verify the authenticity and integrity of the boot code is to include Original Equipment Manufacturer (OEM)-related information in the system. For instance, the public key of the system owner can be used to verify the ownership of the boot code at the startup; in this way, the owner's public key shall be embedded in a Read Only Memory (ROM), and it shall be used at the startup of the system to verify the signature of the boot code that has been previously signed with the corresponding private key. This approach implies that all devices use the same public key. As reported in [1], "storage of the public key for the root of trust can be problematic; embedding it in the on-SoC ROM implies that all devices use the same public key. This makes them vulnerable to class-break attacks if the private key is stolen or successfully reverse-engineered". The same concept can be applied to symmetric keys in case they are used to encrypt the boot code. Starting from these issues, the main goals of our secure boot concept are:

– Avoid the physical storage of secret keys: this eliminates the possibility of disclosure of secret keys stored in memories.

– Unique secret key for each physical device: if the secret key of a device is discovered, it cannot affect the security of the other devices as each key generated by the PUFs is unique.
– Possibility to modify OEM public key stored in the device: in case of disclosure of the OEM private key, the OEM should be able to update its public key stored in the device.
– Unpredictability of the secret keys for each physical device: this resolves issues related to the trustiness of the electronic manufacturers and supply chain.

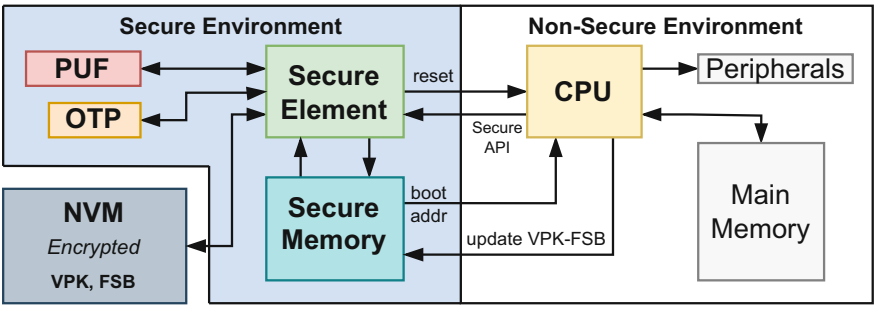


**Fig. 1.** Concept architecture of the secure system. NVM is a Non-Volatile Memory. OTP is a One-Time Programmable memory. VPK is the Vendor's Public Key. FSB is the First-Stage Bootloader of the CPU.

The system we propose is illustrated in Fig. 1, and it is composed of three main parts:

– Secure Environment (S-Env): provides PUF-based encryption and authentication (hardware fingerprint) and can reset the Non-Secure part. We expect the need of an OTP memory to store the PUF challenge plus auxiliary information to retrieve the corresponding response (e.g. redundancy of correction codes, if needed).
– Non-Secure Environment (NS-Env): is a standard CPU-based SoC whose boot sequence is regulated by the Secure part.
– A Non-Volatile Memory (NVM): contains the encrypted Vendor Public Key (VPK) and the encrypted First-Stage Bootloader (FSB) for the CPU.

The S-Env exposes the following Secure Application Program Interface (SAPI) to the NS-Env, based on the status of the OTP memory:

– WriteNVM: this operation initializes the S-Env. It can be issued only if the OTP memory is not written.
– UpdateNVM: the NS-Env can request an update of the NVM employing Publick Key Authentication using the VPK. It can be issued only if the OTP memory has been written.

The boot sequence of the system depends on the S-Env status. As long as the S-Env is not initialised (fresh OTP memory), the Secure Element (SE) copies the content of the NVM into the Secure Memory (SMem) and wakes up the CPU. No verification is performed in this procedure, and the boot is not secure.

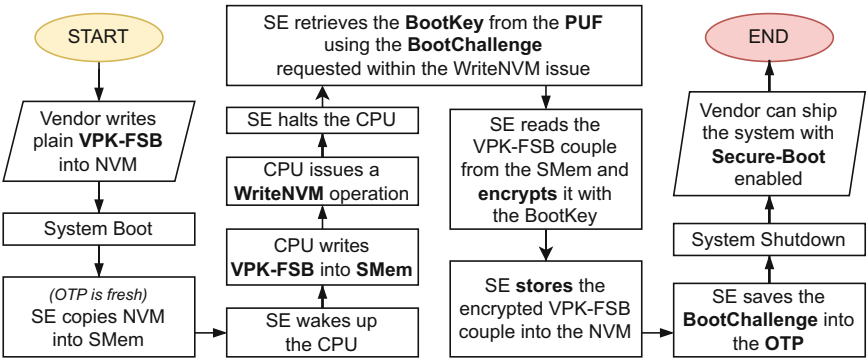


**Fig. 2.** Secure environment initialisation procedure

*Secure Environment Initialisation.* The vendor of the system can initialise the S-Env by performing the following procedure, illustrated in Fig. 2:

1. Vendor prepares the NVM with the VPK-FSB couple and a bootable code executed by the CPU to perform the initialisation sequence.
2. At system boot, the SE will perform a non-secure boot sequence (the OTP is fresh) by coping the NVM content at the boot address of the SMem and waking up the CPU.
3. The code executed by the CPU prepares the SMem with the VPK-FSB couple and the Boot Challenge (BC). The latter can be hard-coded into the initialisation code or determined at run-time. Then, the CPU issues a WriteNVM request to the SE.
4. After halting the CPU, the SE retrieve the Boot Key (BK) from the PUF by providing the BC.
5. The SE encrypts the VPK-FSB couple found in the SMem with the BK using an encryption algorithm (e.g. Advanced Encryption Standard) and stores it in the NVM with a hash digest used for integrity check during secure boot. Depending on the computational capabilities of the SE, the vendor may employ a Digital Signature Algorithm (DSA) or a Hash-based Message Authentication Code (HMAC) to ensure the integrity and authenticity of the FSB code.
6. Finally, the SE saves the BC into the OTP (with any other required parameters related to PUF implementation), and the S-Env can be considered initialised.

7. From now, the WriteNVM operation cannot be issued any more, and the vendor can ship the system with Secure-Boot enabled.

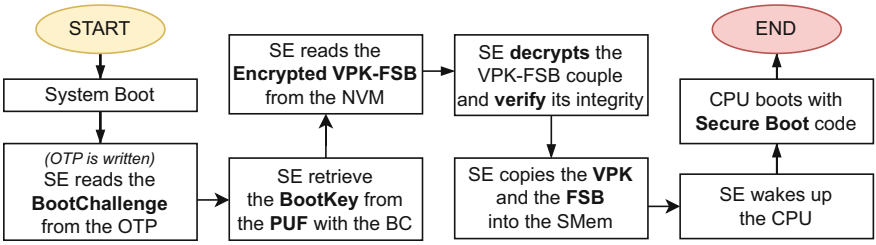


**Fig. 3.** Flow diagram describing the secure-boot sequence

*Secure Boot Sequence.* When the S-Env is initialised, the boot sequence includes the PUF-based verification of the NVM, illustrated in Fig. 3:

1. At system boot, the SE reads the BC from the OTP and retrieves the BK by challenging the PUF.
2. The SE reads the encrypted VPK-FSB couple from the NVM, decrypts it with the BK using an encryption algorithm (e.g. Advanced Encryption Standard) and verifies its integrity and authenticity using DSA or HMAC.
3. Finally, the VPK and the FSB are copied into the SMem, and the CPU is woken up.

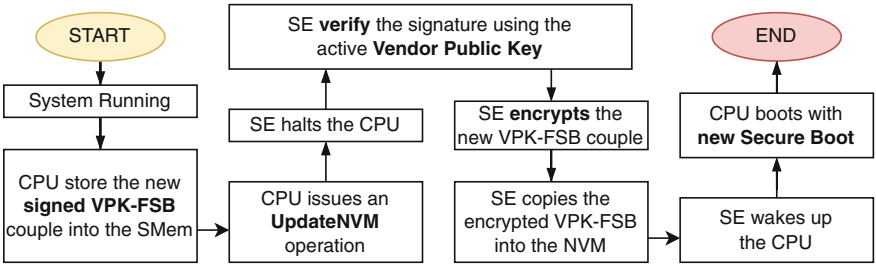


**Fig. 4.** Flow diagram for updating the NVM

*Update NVM Sequence.* During normal operation, the CPU can issue an UpdateNVM operation to update the VPK or the FSB contained in the NVM, as illustrated in Fig. 4:

1. While the system is running, the CPU prepares a new VPK-FSB couple and signs it with the active private key.

2. The CPU store the signed content into the SMem and issues the UpdateNVM operation.
3. The SE halts the CPU and verifies the signature in the SMem using the active VPK.
4. If successful, the SE encrypts the new VPK-FSB couple and stores it in the NVM.
5. Finally, the CPU is woken up, and the new Secure Boot process is performed.

The proposed PUF-based Secure Boot procedure and SAPI avoid the physical storage of secret keys and allow to have unpredictable and unique secret keys for each physical device. In addition, the update sequence allows modifying the OEM VPK in case of private-key class-break attacks to limit the number of compromised devices.

## 3 Implementation on FPGA

To prototype the proposed PUF-based Secure Boot procedure, we implemented a System-on-Chip (SoC) on a Xilinx ZCU106 FPGA Board that emulates the architecture of Fig. 1.

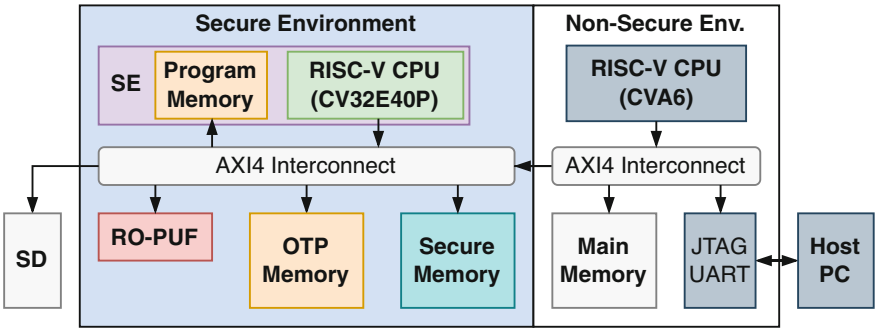


**Fig. 5.** SoC implemented on a Zynq Ultrascale+ FPGA.

As illustrated in Fig. 5, the system is divided into a Non-secure Environment and a Secure Environment. The former (that emulates a general-purpose application system) includes a RISC-V CPU (i.e., the CVA6 64-bit processor [13]) connected through an AXI4 interconnect to its Main Memory (DDR4 controller) and the UART JTAG peripheral provided as Xilinx IPs. The latter implements the SE using a smaller RISC-V CPU (i.e., the CV32E40P 32-bit processor [4]) with its Program Memory. On-chip memory is used to emulate the SMem shared between the two environments and the OTP of Fig. 1. The VPK-FSB couple is stored encrypted on the external SD memory (NVM in Fig. 1). The PUF we implemented is a Ring Oscillator PUF (RO-PUF), generating a response of 325 bits using a challenge of 220 bits. To improve the reliability of the responses,

the output is corrected using an Error Correcting Code (ECC) based on the Reed-Muller code, and we used the VHDL implementation published in [6]. The Red-Muller ECC needs a helper generated during the initialisation procedure and stored on the OTP memory. To generate the AES-CBC-128 key, the SHA256 algorithm is applied to the PUF response and truncated to 128 bits. While to ensure the integrity and authenticity of the VPK-FSB couple, the ECDSA-256 algorithm is used.

## 4 Conclusions

Our research paper presents a novel implementation of a Secure Boot process, which leverages PUF technology to eliminate the requirement of storing secret keys in physical storage. PUF technology enables the generation of unique and unpredictable secret keys for each individual device. Furthermore, our updating process streamlines the ability to modify VPK in order to mitigate the effect of class-break attacks in case of disclosure of the vendor's private key. As a proof-of-concept, we implemented the proposed PUF-based Secure Boot procedure on a Xilinx ZCU106 FPGA board, which includes a 32-bit RISC-V-based secure environment exposing a Secure API for the 64-bit RISC-V application-class processor.

**Acknowledgments.** This work has been partially supported by the European Union within the Horizon 2020 Research and Innovation Programme "European Processor Initiative-Specific Grant Agreement 2" (EPI-SGA2) under Grant 101036168.

## References

1. ARM security technology—Building a secure system using Trust Zone technology (2005–2009). https://developer.arm.com/documentation/PRD29-GENC-009492/c
2. Baldanzi L, Crocetti L, Di Matteo S, Fanucci L, Saponara S, Hameau P (2019) Crypto accelerators for power-efficient and real-time on-chip implementation of secure algorithms. In: 2019 26th IEEE international conference on electronics, circuits and systems (ICECS), pp 775–778. https://doi.org/10.1109/ICECS46596.2019.8964731
3. Di Matteo S, Lo Gerfo M, Saponara S (2023) Vlsi design and fpga implementation of an ntt hardware accelerator for homomorphic seal-embedded library. IEEE Access 11:72498–72508. https://doi.org/10.1109/ACCESS.2023.3295245
4. Gautschi MEA (2017) Near-threshold risc-v core with dsp extensions for scalable iot endpoint devices. IEEE Trans Very Large Scale Integr (VLSI) Syst 25(10):2700–2713. https://doi.org/10.1109/TVLSI.2017.2654506
5. Haj-Yahya J, Wong MM, Pudi V, Bhasin S, Chattopadhyay A (2019) Lightweight secure-boot architecture for risc-v system-on-chip, pp 216–223 (2019). https://doi.org/10.1109/ISQED.2019.8697657, https://www.scopus.com/inward/record.uri?eid=2-2.0-85065164064&doi=10.1109%2fISQED.2019.8697657&partnerID=40&md5=dbd40305c40b567a1f6089561f3d8863 cited by: 17; All Open Access, Green Open Access

6. Liguori P, Reed-muller-decoder. https://github.com/piliguori/Reed-Muller-Decoder
7. Liu, Y., Briones, J., Zhou, R., Magotra, N.: Study of secure boot with a fpga-based iot device. vol. 2017-August, p. 1053 - 1056 (2017). https://doi.org/10.1109/MWSCAS.2017.8053108, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85034084065\&doi=10.1109\%2fMWSCAS.2017.8053108\&partnerID=40\&md5=9cba3d9340f4b2807acc0ab560f9da3fcitedby:24
8. Nannipieri P, Crocetti L, Di Matteo S, Fanucci L, Saponara S (2023) Hardware design of an advanced-feature cryptographic tile within the european processor initiative. IEEE Trans Comput 1–14. https://doi.org/10.1109/TC.2023.3278536
9. Nannipieri P, Matteo SD, Baldanzi L, Crocetti L, Zulberti L, Saponara S, Fanucci L (2022) Vlsi design of advanced-features aes cryptoprocessor in the framework of the european processor initiative. IEEE Trans Very Large Scale Integr (VLSI) Syst 30(2):177–186. https://doi.org/10.1109/TVLSI.2021.3129107
10. Sabt M, Achemlal M, Bouabdallah A (2015)Trusted execution environment: What it is, and what it is not 1:57–64. https://doi.org/10.1109/Trustcom.2015.357, https://www.scopus.com/inward/record.uri?eid=2-s2.0-84967164163&doi=10.1109%2fTrustcom.2015.357&partnerID=40&md5=358ca116c45a82c981b7654c287b8ef6, cited by: 275; All Open Access, Green Open Access Access, Green Open Access
11. Shamsoshoara A, Korenda A, Afghah F, Zeadally S (2020) A survey on physical unclonable function (puf)-based security solutions for internet of things. Comput Netw 183:107593. https://doi.org/10.1016/j.comnet.2020.107593, https://www.sciencedirect.com/science/article/pii/S1389128620312275
12. Wang W, Chen Q, Yin Z, Srivastava G, Gadekallu TR, Alsolami F, Su C (2022) Blockchain and puf-based lightweight authentication protocol for wireless medical sensor networks. IEEE Internet Things J 9(11):8883–8891. https://doi.org/10.1109/JIOT.2021.3117762
13. Zaruba F, Benini L (2019) The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology. IEEE Trans Very Large Scale Integr (VLSI) Syst 27(11):2629–2640. https://doi.org/10.1109/TVLSI.2019.2926114
14. Zhang J, Qu G (2020) Physical unclonable function-based key sharing via machine learning for iot security. IEEE Trans Ind Electron 67(8):7025–7033. https://doi.org/10.1109/TIE.2019.2938462