DOI: 10.15439/2023F6946
ISSN 2300-5963 ACSIS, Vol. 35

IoTrust - a HW/SW framework supporting security core baseline features for IoT

Mateusz Korona[*], Bartosz Zabołotny[†], Fryderyk Kozioł [‡], Mateusz Biernacki[§],
Radosław Giermakowski[¶], Paweł Rurka[‖] Marta Chmiel[**], Mariusz Rawski[††]
[*]0000-0002-9718-9684, [†]0000-0002-3364-1766, [‡]0000-0002-6312-2406, [§]0009-0008-7765-800X,
[¶]0009-0001-3997-4369, [‖]0009-0000-5014-8255, [**]0000-0002-9718-9684, [††]0000-0002-7489-0785,
Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
Email:{mateusz.korona, bartosz.zabolotny, fryderyk.koziol, mateusz.biernacki,
radoslaw.giermakowski, pawel.rurka, marta.chmiel, mariusz.rawski}@pw.edu.pl

***Abstract*—The rapid growth of the Internet of Things has significant security implications. In the current IoT security landscape, many institutions and entities are defining security requirements, but no industry-wide standard has been agreed upon. There are solutions in the present state-of-the-art that fulfill a subset of secure IoT device requirements, but none adheres to all of them. However, the existing technologies introduced by those solutions could be combined to create a design framework which provides security baseline features to support requirements of a secure IoT device. In this paper, a configurable and comprehensive hardware-software security framework is proposed, that, when applied in the process of designing System on Chip for IoT, will ensure its cybersecurity by providing security core baseline features. The proposed sollution is CPU-agnostic, in the sense that no assumptions are made about the CPU's support for privilege levels, memory protection schemes, or any security mechanisms.**

I. Introduction

IoT devices are becoming an increasingly important aspect of our lives and can be sensed everywhere around us.

Due to the inherent characteristics of IoT devices, data is continuously transmitted, processed, and stored in the cloud. Studies have indicated that many IoT devices that have been compromised lack adequate security measures. IoT security is not just device security, as all elements need to be considered, including the device, cloud, mobile application, network interfaces, software, use of encryption, use of authentication, and physical security. Recent research directions in IoT focus on addressing these challenges and improving the performance and security of IoT systems [1].

Much research focuses on software, network, and cloud security; however, hardware security in these devices has been overlooked. Although software-based solutions are less expensive to implement and update, they have their limitations and are also more vulnerable to attacks. Hardware-based solutions may be more expensive and time-consuming to implement, but integrating hardware security can significantly strengthen the system's defenses against attackers and in the long run, this type of solution is better positioned to protect sensitive communications and personal data from exposure.

This work was funded by The Polish National Centre for Research and Development under project No. CYBERSECIDENT/456446/III/NCBR/2020.

Many renowned institutions have already come forward with their security guidelines for developers, distributors, and users. Among the first was the National Institute of Standards and Technology (NIST), that in [2] has defined an Internet of Things (IoT) device cybersecurity capability core baseline, which is a set of device capabilities generally needed to support common cybersecurity controls that protect an organization's devices as well as device data, systems, and ecosystems. This core baseline provides organizations a starting point to use in identifying the device cybersecurity capabilities for new IoT devices they will manufacture, integrate, or acquire.

In [3] the authors provide an overview of security guidelines for IoT proposed by various organizations and evaluate some of the existing technologies applied to ensure IoT security against these guidelines. In the paper, recommendations proposed by selected government organizations, international associations, and advisory groups are gathered and compiled into a set of the most common and important considerations, divided into eight categories. Then the authors chose a number of representative examples from IoT security technologies and evaluated them against these criteria. Conclusions captured in that paper show that there is no exhaustive and CPU-agnostic solution. While none of the examined solutions fulfill all recommendations on their own, the existing technologies introduced by those solutions could be combined to create a design framework that satisfies all the requirements of a secure IoT device.

In this paper the concept of an IoTrust framework has been proposed. This hardware-software solution, when applied in the process of designing System on Chip (SoC) for IoT, will ensure its cybersecurity by providing security core baseline features. The proposed framework architecture assumes the combination of the mechanisms from the area of Hardware Root of Trust (HWRoT), Trusted Execution Environment (TEE), and Trusted Computing in order to ultimately create a configurable and comprehensive solution ensuring the security of IoT nodes.

II. State of the Art

The security of IoT devices is an area of active research due to unsatisfactory levels of safety and the immense range of applications. The constrained resource nature of many IoT devices increases the challenge of an all-in-one solution. That is why there are many solutions that solve pinpointed areas of the SoC, but the security of the entire device is still a novel topic which has yet to have an industry-wide accepted solution.

A. Key enabling technologies

Security architectures defined for traditional embedded systems are also currently used in IoT devices, solving some security issues. However, further improvements to these architectures are necessary to address new varieties of device vulnerabilities in the IoT ecosystem. Trustworthy computing is a major challenge in the field of cybersecurity.

1) Trusted Execution Environments: Solution which provides a secure environment for applications to run, regardless of the security in the rest of the system. TEEs are complex systems that consist of both hardware and software components and offer an enhanced execution environment [4]. TEE is a tamper-resistant computing environment running a separation kernel that guarantees the authenticity of program code, integrity of crucial system assets (processor registers, secured memory), and confidentiality of code and data stored in persistent memory [4]. Additionally, a TEE is useful in providing authentication and identification of the system. To the outside world, TEE is a module that guarantees isolation between secure and non-secure environments for both code and data.

2) Hardware Root of Trust: A key technical challenge for TEEs is ensuring trust, meaning that the system behaves as expected by the user. To address this challenge, there has been significant support for the use of hardware-based root-of-trust (HRoT) implementations to establish trust in secure computing. Hardware RoTs are preferred over software RoTs due to their immutability, smaller attack surfaces, and more reliable behavior. They can provide a higher degree of assurance that they can be relied upon to perform their trusted functions [5].

3) Physically Unclonable Functions: A Physically Unclonable Function (PUF) is a physical random function that typically displays a unique challenge-response behavior for each of its instances. The response to a given challenge is randomly generated based on the intrinsic physical properties of the hardware in which it is embedded. Recently, PUFs have been proposed as key components in cryptographic mechanisms and security architectures [6]. They can be used for device identification and authentication, binding software to hardware in a platform, securely storing cryptographic secrets and designing secure protocols.

B. Existing solutions

Trusted computing solutions for IoT devices are essential for ensuring the security and integrity of IoT systems. To address this challenge, a number of solutions have been proposed. Some of them are already mature and currently in use for IoT devices, and some are emerging concepts that might bring new quality to the topic.

The evaluation of representative IoT security technologies against criteria presented in [3] shows that while there are solutions with the potential to meet all these recommendations, no solution currently addresses all requirements in an out-of-the-box capacity. This leaves room for further research in this field.

Here, we briefly describe a few example solutions.

1) ARM TrustZone: A security extension provided by ARM for both application processors (Cortex-A family) and microcontrollers (Cortex-M family) [7]. It is based on a TEE concept. TrustZone divides the system into secure and non-secure environments by providing two virtual processors with hardware-based access control. Memory isolation and a special processor mode dedicated to monitoring (the secure monitor) ensure complete separation of the two execution environments in hardware.

TrustZone offers a comprehensive security solution, but it requires a good understanding of the framework, creative implementation, and support from external IPs. It is important to note that TrustZone is designed for ARM infrastructure and relies on additional hardware such as CryptoCell and applications to do so. TrustZone alone is not an off-the-shelf, ready-to-use solution.

2) Intel SGX: A set of CPU instructions that enable the creation of isolated software containers called enclaves [8]. These enclaves provide a secure environment for a program's code, data, and stack through hardware-based access policy control and memory encryption. This isolation protects the program from other processes, even those with higher privilege levels. From a hardware perspective, Intel SGX isolates Processor Reserved Memory (PRM) and protects it against all memory accesses from outside an enclave. This includes access attempts by the kernel, hypervisor, system management mode, and DMA accesses requested by peripherals.

3) Keystone: An open-source framework designed for creating TEE environments based on an unmodified RISC-V architecture [9]. It uses RISC-V Physical Memory Protection (PMP) and the programmable machine mode (M-Mode) to implement a memory protection scheme. The Trusted Security Monitor (SM) is proposed at the M-Mode level and is responsible for managing secure hardware handling and context switching between enclaves. The SM should be executed entirely from on-chip memory and satisfies typical TEE requirements such as memory isolation and code/configuration attestation. Keystone does not provide direct resource management; this responsibility falls on the secure enclave application developer. Also, several platform requirements are listed by the authors, including support for a trusted boot process, an unique authentication key dedicated to this process, and a hardware source of randomness. Keystone can be a good starting point for securing an IoT device, but it is not sufficient on its own.

4) OpenTitan: An open-source Hardware Root of Trust implementation endorsed by leading non-profit, academic, and commercial organizations [10]. As an open-source project, its sources are available online for inspection by the broader community, which should improve its security. While OpenTitan's core is still under development, and several features are missing from its early-stage top-l el, its creators' intentions are well-documented. In its complete form, OpenTitan should be a robust solution for securing various systems as a HWRoT module that supports secure boot procedures and implements miscellaneous cryptographic primitives.

5) CURE: A security architecture providing Trusted Execution Environments with different types of enclaves: subspace enclaves provide vertical isolation at all execution privilege levels, user-space enclaves provide isolated execution to unprivileged applications, and self-contained enclaves allow isolated execution environments that span multiple privilege levels. CURE's protection mechanisms are based on new hardware security primitives on the system bus, the shared cache, and the CPU. It also enables the exclusive assignment of system resources, such as peripherals, CPU cores, or cache resources, to a single enclave [11]. The authors assume the CPU supports privilege levels to separate user space from the more privileged kernel space through virtual address spaces using a MMU. Moreover, it is assumed that the system performs a secure boot on reset, with the first bootloader stored in CPU Ready-Only Memory (ROM) and verifying the firmware through a chain of trust.

III. Concept of the IoTrust framework

In this work, we present a security framework called IoTrust that addresses security issues in a customizable way. The solutions presented are CPU-agnostic, meaning no assumptions are made about the CPU's support for privilege levels, memory protection schemes, or any security mechanisms.

A. Threat model

The presented framework focuses on securing code integrity, control flow integrity, and confidentiality and integrity of secrets of an application running on an IoT device installed in the field. The secrets to be protected include encryption keys, certificates, and hashes, as well as the application's sensitive data. The IoTrust architecture's trusted computing base consists of the system's on-chip hardware components as well as a dedicated hypervisor software component. While the source of data in on-chip memory is assumed to be correct, the framework does not trust off-chip memory, the operating system, the applications, or the physical protection provided by the device manufacturer. Side-channel attacks, cloud security, and network security are beyond the scope of this work. Potential attacks include cold boot attacks, physical code injection, and compromising the hypervisor.

B. IoTrust architecture

Fig. 1 shows the architecture of a sample SoC system for an IoT device based on a standard CPU, system bus fabric, typical off-chip memory blocks and input/output devices with additional specialized components of the IoTrust framework.

The IoTrust framework is intended to be a configurable SoC framework for IoT devices that leverages TEE environment concepts. A set of developed IP Cores is proposed, which enable the implementation of a HWRoT, proxy modules that filter interfaces that are connected to off-chip components, a secure DMA that encrypts based on per-enclave cryptographic keys and, a CPU agnostic module that allows compartmentalization of software execution space. The proposed solution can be adapted for IoT implementations using both FPGA (Field Programmable Gate Array) and ASIC (Application Specific Integrated Circuit) technologies.

The software part of the framework consists of the **Security Manager Software** (SM-SW). It is responsible for managing enclaves and their lifecycle while acting as a hypervisor. It allocates processor time to enclaves and queues enclaves that are waiting for CPU execution time. Using emulated software interrupts, enclaves are provided with a secure way to interact with the rest of the system through the SM-SW API and data exchange between enclaves is enabled. The SM-SW module also includes interrupt handling of interrupts generated by the Security Manager Hardware (SM-HW) module during active enclave switching. This procedure ensures a safe and secure switch between a potentially untrusted enclave and the hypervisor code, which has full access to all system components. For this implementation, the **enclave context** is defined as the extension of the processor's context (the state of the CPU registers used by the executed program) by adding the state of API registers included in the SM-HW.

Furthermore, the SM-SW ensures proper configuration of the hardware modules of the framework at the boot time. Before starting enclaves, it writes definitions of the privileges enforced by the hardware, such as the interrupt filtering or virtual address spaces, to SM-HW registers.

To guarantee a safe enclave switching procedure and successful preemption even if the CPU runs malicious code, SM-SW programs a **handshake sequence** based on handler execution and a watchdog timeout.

The software part includes libraries that provide trusted Application Programming Interfaces (APIs), enabling the implementation of typical cybersecurity mechanisms and modules that implement functionalities outlined in the aforementioned core baseline requirements. The solution can provide a comprehensive, configurable software/hardware framework for building a trusted TEE runtime environment together with hardware specific to IoT security solutions.

The hardware part of the IoTrust framework consists of the following main parameterizable modules that are added at the SoC level.

The **Security Manager Hardware** together with the Security Manager Software enables the implementation of Trusted Computing concepts by creating enclaves where code is executed and ensuring their physical isolation (each enclave has access to dedicated hardware resources), which, with the proper use of the HWRoT module, allows for the implementation of a state-of-the-art TEE environment. It manages transactions between CPU and system bus. Additionally, in-

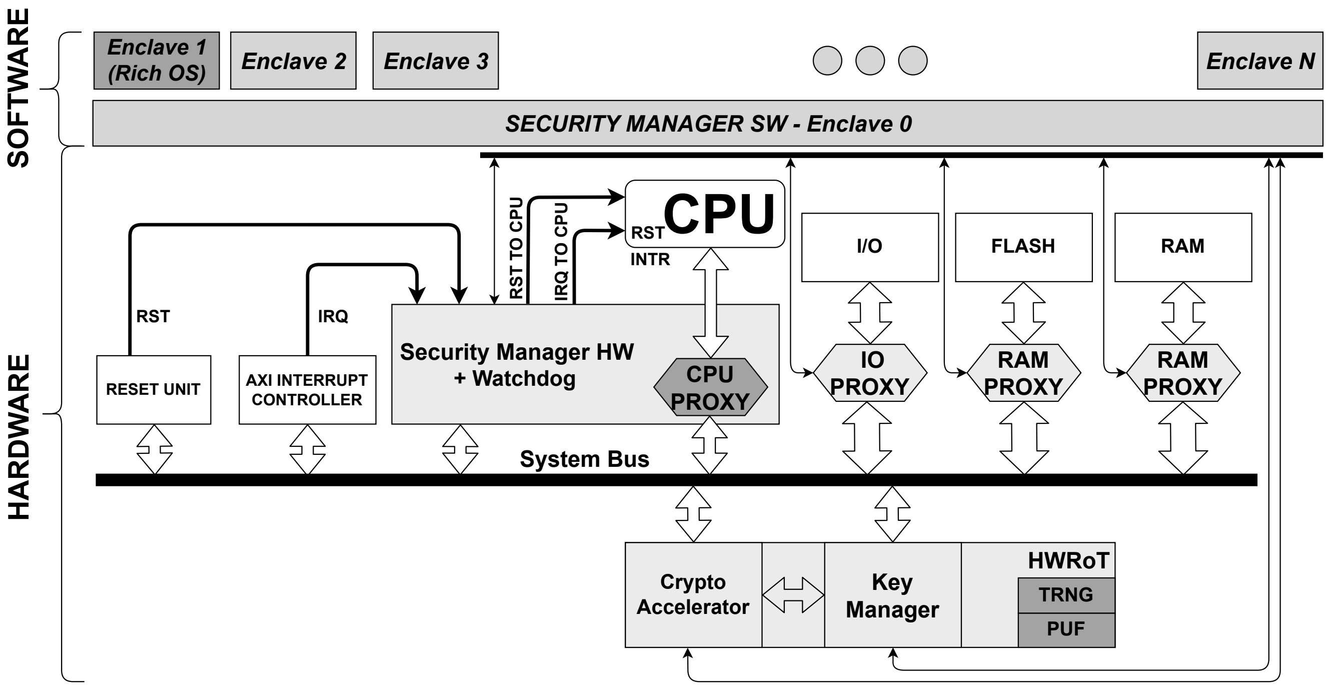

Fig. 1. Block level diagram of an example SoC integrating with the IoTrust framework

formation about the currently active enclave is passed to other IoTrust framework blocks to ensure synchronized isolation of processes from system devices and enable memory access (encryption/decryption) associated with the keys tied with the currently running enclave. Configuration changes and access violations are reported using a cybersecurity event logging module and the created logs are sealed to ensure their confidentiality and integrity when read by an authorized entity.

The **Hardware Root of Trust** (HWRoT) acts as an anchor of trust in the system and provides secure hardware implementations of necessary cryptographic algorithms. This component consists of hardware modules that provide security functions necessary to ensure trust within a platform (such as confidentiality, integrity, verification, authorization, secure storage, and updating). Its essential traits are immutability and predictability - it always behaves in the same way under known conditions. A hardware implementation enables meeting these conditions. HWRoT is treated as an inherently trusted element of the system. Thanks to the security services it offers, it can verify the correctness of subsequently launched software modules during the runtime of a Rich OS or other kernels. In this way, trust can be propagated within the platform and, the so-called Chain of Trust is formed, with HWRoT as the first element and the operating system or application running in it as the last. The secure system startup process including the creation of the Chain of Trust is called secure boot.

The **RAM Proxy** handles accesses to an external RAM with data protection. The task of the RAM Proxy module is to secure and manage accesses to external RAM. It is an essential component that mediates between the SM-HW and the memory controller during data exchange. Additionally, it works closely with the HWRoT module for data encryption and decryption.

IV. Implementation

The IoTrust framework's components have been developed using Verilog HDL and C/assembler. An example SoC based on the Xilinx Microblaze soft CPU and Vivado system platform has been designed as a proof of concept.

A Trusted Execution Environment of the IoTrust framework is a secure and isolated execution area within a computing unit that provides the authenticity of executed code, integrity of resources (such as CPU registers, memory, or input/output devices), and confidentiality of code, data, and states of non-volatile memory. In the IoTrust framework a TEE is implemented using the concept of enclaves.

An **enclave** is defined as a secure runtime environment managed by the Security Manager software and hardware, along with its metadata (a set of hardware access permissions, checksums, and digital signatures for the code) and an isolated address space where verified application instances are launched. Each enclave running on the processor operates in a separate virtual address space. Access to peripherals is limited by the Security Manager Hardware and is configurable per enclave.

A. IoTrust Hardware

1) Security Manager Hardware: The Security Manager Hardware module (Fig. 2) is directly connected to the CPU and is responsible for implementing Trusted Computing features. It is composed of:

- Enclave switch control – a module that monitors the CPU instruction bus to detect the sequence of instructions indicating enclave switch. The sequence can be programmed in the register file. The CPU Proxy is instructed to change the enclave context when the sequence is detected.
- CPU Proxy – a module responsible for hardware translation of transaction addresses from the virtual space seen by a given enclave to the physical space of the system bus. Transactions that violate access rights are rejected, and security breach logging processes are started. The module labels each AXI transaction with the number of the currently active enclave using AXI USER signals. This information is used by other IoTrust framework blocks to ensure isolation of enclaves from each other and enable memory access by encryption/decryption of its content using the keys tied with the currently running enclave.
- Interrupt Proxy – system interrupts are intercepted and subsequently forwarded to the computing unit only if they are authorized to be handled by the currently executing enclave. The Current Enclave register and the Watchdog interrupt mechanisms enable run-time security control of the system and detect anomalies. If the designated enclave fails to handle a specific interrupt within the appropriate time, the Watchdog module signals a system malfunction due to an error or a cyber attack. The event is then reported in a sealed log, and the system is reset.

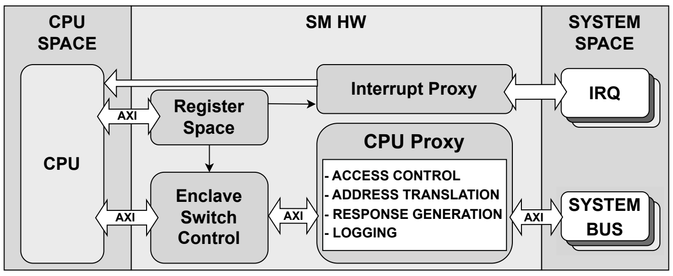

Fig. 2. Block level diagram of the SM HW module

2) Hardware Root of Trust: The Hardware Root of Trust of the IoTrust platform (Fig. 3) consists of three main components. The first one is a read-only memory for the first bootloader, which performs the initial system initialization and does the initial configuration of the HWRoT blocks during the secure boot procedure. The next component is the crypto accelerator block, which implements necessary cryptographic primitives (including lightweight algorithms suitable for hardware resource-constrained devices) and includes a DMA unit for efficient data operations. This module assists in the encryption of RAM for individual enclaves. The last component is the **secret top** block, which manages cryptographic keys and other secrets (such as physical and logical device identifiers, owner identity). It also generates secure cryptographic random sequences (to generate cryptographic keys within the HWRoT itself) and includes a Physical Unclonable Function block, which provides a unique fingerprint for each device.

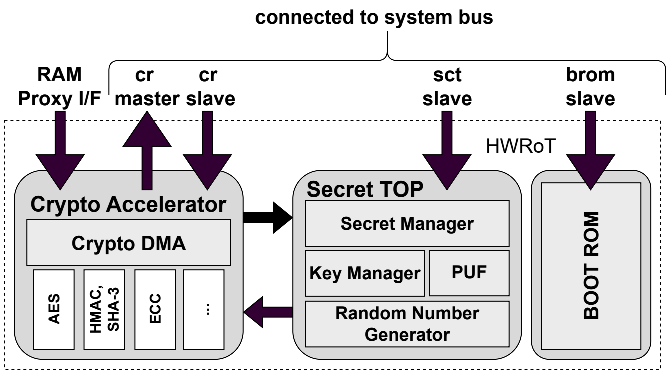

Fig. 3. Block level diagram of the HWRoT module

3) RAM Proxy: RAM Proxy is a component which connects system and the RAM. It intercepts memory accesses and is responsible for data protection. At first, appropriate data block is read from the memory and passed to HWRoT with enclave identifier (for cryptographic key selection) to be decrypted. If the requested operation is read, the decrypted data block is transmitted on the system bus. In the case of write operation, the corresponding data fragment is replaced, and the data block is then re-encrypted in HWRoT, and passed to the memory controller. Module has cache buffer that stores recently read memory blocks and increases the system's efficiency, because it allows to reduce the number of interactions with off-chip memory and the HWRoT.

4) IO Proxy: Allows hardware-based control of interface access in the SoC. For example, a local JTAG debug interface, Bluetooth, or Ethernet. It can filter transactions to and from peripherals. It contains a register that must be accessed by the SM-SW before usage due to hardware access restrictions when the peripheral is not enabled.

B. IoTrust Software

The SM-SW is functionally split into several modules. The Enclave Manager contains the main loop and is responsible for enclaves' management. Memory manager allocates/deallocates memory regions (*memory slices*) and is responsible for their safe clearing. It translates between virtual space and physical system bus addresses (Fig. 4). The hypervisor implements dedicated mechanisms for scheduling, safe switching of the enclaves, and data exchange between enclaves.

V. Synthesis results

Synthesis and implementation were performed using Vivado tools version 2022. The results presented are for the Trenz evaluation board TE0712 [12] equipped with a Xilinx Artix-7 XC7A200T FPGA. Table I presents the resource utilization of the entire system by assessing the number of LUTs, Slice, DSPs and block RAMs.

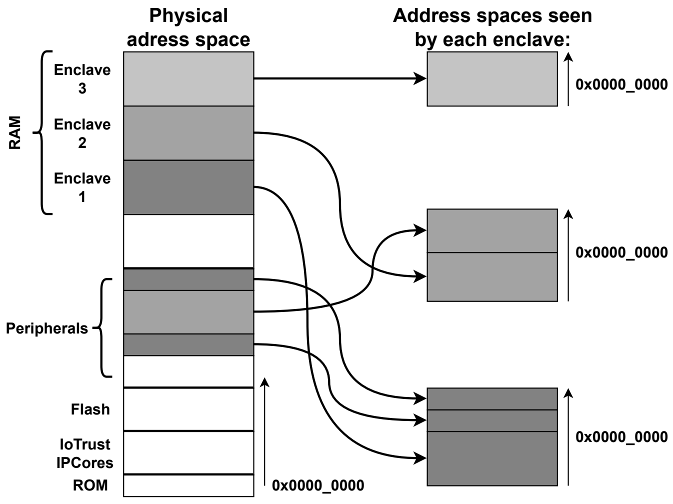

Fig. 4. Virtual address spaces of the enclaves are mapped onto the system's physical address space.

TABLE I
UTILIZATION RESULTS

	Used	Available	Util [%]
Slice LUT	59719	133800	44.63
LUT as Logic	58105	134600	43.19
LUT as Memory	2064	46200	4.47
Slice Registers	31023	269200	11.52
Slice	18363	33650	54.57
Block RAM	41	365	11.23
DSPs	4	740	0.54

The article presents the Proof of Concept of the IoTrust framework. For this reason, the target solution may be optimized for utilization, power consumption, or frequency. The results presented do not include the HWRoT module. Considering the number of logic cells used, the solution can be classified as a lightweight solution.

The power estimation analysis indicates, that the whole SoC consumes about $1.5\ W$ of power. Since the DDR memory controler consumes about $1\ W$ of power, the rest of the system components use $0.5\ W$ of power.

VI. CONCLUSIONS

The IoT ecosystem presents new security challenges beyond traditional data security. There is a need for IoT security guidelines, and many organizations worldwide have proposed recommendations to help ensure secure IoT infrastructure. While device designers and vendors have their own proprietary solutions that address some issues, they fall short in others. Evaluation of representative examples of IoT security technologies shows that while there are solutions with the potential to meet all recommendations, none currently do so in an out-of-the-box capacity.

In this paper, we proposed the concept of the hardware-software security framework that, when applied in the process of designing System on Chip for IoT device, will ensure its cybersecurity by providing security core baseline features. The proposed IoTrust framework consists of custom hardware IP Cores designed in Verilog HDL as well as C/assembler software procedures that can be included in SoC design, enabling the combination of the mechanisms from the area of Hardware Root of Trust, Trusted Execution Environment and Trusted Computing. This ultimately creates a configurable and comprehensive solution ensuring the security of IoT nodes.

The solution discussed does not make any assumptions about the CPU's support for privilege levels, memory protection schemes, or security mechanisms and is therefore CPU-agnostic.

A proof-of-concept implementation has been demonstrated where the IoTrust framework has been applied to SoC based on MicroBlaze soft-processor. As the prototype platform the Trenz evaluation board TE0712 equipped with a Xilinx Artix-7 XC7A200T FPGA has been used. Example execution scenarios have been included, that demonstrate basic functionalities of proposed solution.

REFERENCES

[1] P. I. Radoglou Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the internet of things: Challenges, threats and solutions," *Internet of Things*, vol. 5, pp. 41–70, 2019. doi: 10.1016/j.iot.2018.11.003.

[2] M. Fagan, K. N. Megas, K. Scarfone, and M. Smith, "IoT device cybersecurity capability core baseline," tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, may 2020. doi: 10.6028/NIST.IR.8259a.

[3] M. Chmiel, M. Korona, F. Kozioł, K. Szczypiorski, and M. Rawski, "Discussion on IoT Security Recommendations against the State-of-the-Art Solutions," *Electronics*, vol. 10, no. 15, 2021. doi: 10.3390/electronics10151814.

[4] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 57–64, 2015. doi: 10.1109/Trustcom.2015.357.

[5] A. Ehret, E. Del Rosario, K. Gettings, and M. A. Kinsy, "A hardware root-of-trust design for low-power soc edge devices," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, 2020. doi: 10.1109/HPEC43674.2020.9286164.

[6] T. Idriss, H. Idriss, and M. Bayoumi, "A puf-based paradigm for iot security," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 700–705, 2016. doi: 10.1109/WF-IoT.2016.7845456.

[7] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Comput. Surv.*, vol. 51, jan 2019. doi: 10.1145/3291047.

[8] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for cpu based attestation and sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, vol. 13, p. 7, ACM New York, NY, USA, 2013.

[9] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, (New York, NY, USA), Association for Computing Machinery, 2020. doi: 10.1145/3342195.3387532.

[10] "Opentitan - open source silicon root of trust." https://opentitan.org/. Accessed on 30.06.2021.

[11] R. Bahmani, F. Brasser, G. Dessouky, P. Jauernig, M. Klimmek, A.-R. Sadeghi, and E. Stapf, "CURE: A security architecture with CUstomizable and resilient enclaves," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1073–1090, USENIX Association, Aug. 2021. doi: 10.48550/arXiv.2010.15866.

[12] T. Electronics, "TE0712 WIKI." https://wiki.trenz-electronic.de/display/PD/TE0712+Resources, 2023. Accessed on 21.05.2023.